

Applying deep reinforcement learning to minimize flow fluctuations in digital flow control

Essam Elsaed^{a,b,*} , Matti Linjama^a 

^a Innovative Hydraulics and Automation, Tampere University, Finland

^b Faculty of Engineering, Ain Shams University, Egypt

ARTICLE INFO

Keywords:

Digital hydraulics
Reinforcement learning
Neural network
Pressure spikes
Flow fluctuations

ABSTRACT

This work addresses the non-linear and complex optimization challenges in digital hydraulic flow control, where piecewise behaviors and unpredictable interactions make traditional optimization methods impractical for presented system with continuous inputs. The study aims to promote real-time application of artificial intelligence algorithms for system optimization. Most off-highway construction and agriculture equipment use hydraulic valve manifolds, which offer unmatched power density and dynamics, excelling over electro actuators in high-capacity applications. The growing demand for more efficient and accurately controlled autonomous heavy machinery has driven the need for steady and precise flow control systems with reduced pressure drop. However, managing flow and pressure fluctuations when switching valves remains a significant challenge.

The proposed agent effectively mitigates flow fluctuations by interactively refining valve-timing decisions over tens of thousands of possible actions. Validated under approximately 90 % of available conditions and tested against unseen pressure values, the agent achieved a median integrated flow error of less than 0.5 cm³, showcasing its potential for AI-driven optimization in digital hydraulic systems.

1. Introduction

1.1. Challenges for managing flow fluctuations in digital hydraulic systems

Traditional proportional and torque motor valves have been widely employed in high-flow hydraulic systems; however, these technologies suffer from inherent limitations, including slow dynamic response, high cost, sensitivity to contamination, and mechanical complexity. To overcome these drawbacks, Digital Hydraulic (DH) valves have emerged as a promising alternative, offering improved versatility, reliability, and efficiency in flow and pressure control (Sciatti et al., 2024).

Nonetheless, digital hydraulic systems, especially those employing parallel-connected valves in pulse code modulation (PCM) configurations, consistently experience unpredictable flow fluctuations and abrupt pressure spikes, negatively impacting system efficiency (Laamanen et al., 2007). Despite extensive investigation over the past two decades, effective solutions remain limited. Consequently, some researchers have resorted to complex, valve-intensive approaches, such as dense valve manifolds and overburdened configurations (Ozalp et al., 2021). Previous studies attempting to mitigate these problems often

relied on costly or elaborate measures, as summarized comprehensively in the review article by (Lu, 2020), underscoring the urgent need for simpler and more cost-effective strategies.

1.1.1. Binary valve control in digital hydraulics

Digital Hydraulic systems with parallel-connected valves utilize multiple on/off valves, where each valve corresponds to a binary bit, enabling quantized control of the flow rate. For a Digital Flow Control Unit (DFCU) with n valves, the total controlled flow rate Q is given by:

$$Q = \sum_{i=0}^{n-1} u_i \cdot 2^i \cdot q_0. \quad (1)$$

Where u_i is the binary state (1: open, 0: closed) of the i -th valve, and q_0 is the smallest (base) flow unit of the ideal DFCU. PCM in digital hydraulics involves encoding flow rates using binary valve states (ON : $1 \cdot 2^i \cdot q_0$ or OFF : $0 \cdot 2^i \cdot q_0$), with valve capacities following binary progression (e.g., $q_0, 2q_0, 4q_0, \dots$). This method provides high-resolution control with minimal hardware complexity, allowing for rapid selection of the valve combination for the desired flow rate (Valmet, 2016). As an illustrative example, consider a PCM system with

* Corresponding author.

E-mail addresses: essameldin.elsaed@tuni.fi (E. Elsaed), matti.linjama@tuni.fi (M. Linjama).

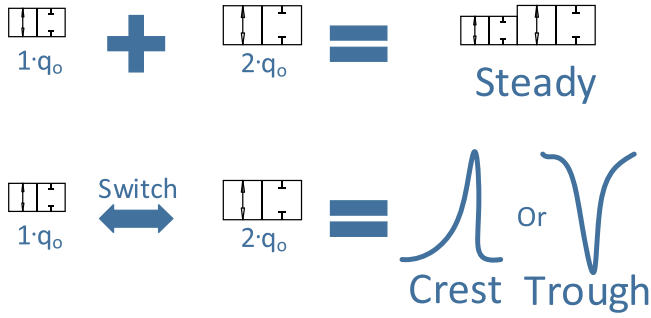


Fig. 1. Peaks (Crest or Trough) during valves transitions in a PCM-Based DFCU ($n = 2$). For illustration purposes only.

6 valves (as analyzed in this study). Each valve i has a flow capacity of $2^i \cdot q_0$. If all valves are open, the corresponding binary state vector is $U = [1, 1, 1, 1, 1, 1]$ and the total flow rate can be calculated as $Q = 1 \cdot 2^0 \cdot q_0 + 1 \cdot 2^1 \cdot q_0 + 1 \cdot 2^2 \cdot q_0 + 1 \cdot 2^3 \cdot q_0 + 1 \cdot 2^4 \cdot q_0 + 1 \cdot 2^5 \cdot q_0 = 63 \cdot q_0$.

1.1.2. Mitigating flow and pressure fluctuations

In PCM digital hydraulic systems, pressure peaks (Fig. 1) are a persistent issue. These peaks occur because on/off valves do not switch simultaneously due to inherent variations in their response times, causing temporary flow rate mismatches during state transitions. This flow rate error leads to pressure peaks (Laamanen et al., 2005). The resulting flow rate summation error $\Delta Q_E(t)$ during valves transitions can be described as $\Delta Q_E(t) = \left(\sum_{j \in O} [Q_j \cdot f_j(t)] - \sum_{j \in C} [Q_j \cdot f_j(t)] \right)_{actual} - Q_{desired}(t)$, where O and C represent the set of indices j corresponding to valves opening or closing. Q_j is the flow capacity of the j -th valve, and $f_j(t)$ is a time-dependent function describing the valve opening or closing behavior.

The state equation for pressure change is given by: $\frac{dp(t)}{dt} = \frac{\beta}{V} \Delta Q(t)$, where β is the bulk modulus, and V is the fluid volume of the studied chamber. Rearranging, the pressure peak caused by timing discrepancies becomes: $\Delta p(t) = \frac{\beta}{V} \int_0^t \Delta Q(t) dt$ (Laamanen et al., 2005). To mitigate pressure peaks, reducing the system β (compressibility creates transient delays (Chapple, 2014)) or minimizing V can help, but these are often constrained by system requirements. Therefore, the primary strategy is to reduce flow mismatches during valve transitions. A traditional approach in digital hydraulics is to use a cost function-based controller that penalizes switching sequences likely to cause pressure spikes. However, this often results in a trade-off between minimizing pressure peaks and maintaining flow accuracy (Laamanen et al., 2007).

1.1.3. Addressing system fluctuations in advanced hydraulic systems

Pressure fluctuations and flow disturbances are challenges across various hydraulic applications. In open-loop independent metering control systems with multiple actuators, mode shifts during operation often introduce significant disturbances, leading to pressure fluctuations (Huova et al., 2020). Similarly, a group of researchers in Industrial and Heavy Machinery (Yang et al., 2023) highlights that poor valve timing contributes to substantial pressure fluctuations. In multi-chamber actuators, unsynchronized valve actions directly cause abrupt pressure spikes. To combat this, research from Linköping University and the Federal University of Santa Catarina (Raduenz et al., 2022) emphasize the importance of precise and coordinated valve control strategies to stabilize pressure and maintain system efficiency.

Digital displacement pumps (Chenggang & Pan, 2024) can benefit from synchronized valve operations to address delay effects, as optimizing low and high pressure valve timing mitigates backflow and efficiency losses. Pneumatic systems similarly benefit from synchronized valve operations, which have been shown to reduce pressure variations

and enhance energy efficiency (Padovani & Barth, 2018). In aerospace applications, especially digital hydraulic actuators, poor synchronization can cause severe pressure peaks or cavitation, highlighting the need for precise timing (Mantovani et al., 2020).

These examples highlight the crucial role of valve timing and synchronization in reducing system fluctuations in transient states. Effective control strategies are key to ensuring stability and efficiency in modern hydraulic systems.

1.2. Literature on optimization approaches to mitigate flow fluctuations

1.2.1. Valve synchronization optimization problem definition

A key approach to optimizing DFCU performance and minimizing flow errors during transitions is to manage valve response times. This involves introducing artificial delays, $t_{Ar,k}$, for the k -th valve (especially faster ones) to compensate for response time differences. The problem can be formalized as:

$$\min_{t_{Ar,k}} \Delta Q_E(t)$$

$$\text{Constraints : } t_{min} < t_{Ar,k} < t_{max} \quad \forall k \quad (2)$$

Where optimization is constrained to discrete delay values bounded by system limitations. The chosen algorithm must be fast, adaptable for real-time adjustments (a direction for future research), and robust against variations in circuit pressure, temperature, and valve characteristics (Elsaed & Linjama, 2024c). Moreover, the nonlinear relationship between artificial induced delay times and system performance requires specialized optimization algorithms. As the system has multiple valves, the problem becomes multi-variable, with the potential for multiple feasible solutions, necessitating efficient techniques to find optimal or near-optimal solutions.

1.2.2. Possible optimization algorithms

To address valve synchronization optimization, various algorithms perform well on similar problems, but none guarantee superior results for this specific case, making algorithm selection crucial, especially with limited computational resources.

One option is to use classical optimization methods (Gradient-Based and Gradient-Free), like gradient descent, nonlinear programming, dynamic programming, or the Lagrange multiplier method. A recent discrete optimization study used dynamic programming in Digital pumps with fast valves, however it faces high computational costs that grow exponentially with the number of states, making it impractical for real-time control in systems with high valve counts (Larsson et al., 2022). These deterministic algorithms are effective for finding exact or approximate solutions in well-structured problems with mathematically defined objectives and constraints, which is *not* achievable in most hydraulic applications.

More promising potential is approximate algorithms, which are classified into problem-dependent heuristics and metaheuristics (MHs) (Karimi-Mamaghan et al., 2022). Heuristics are designed for specific optimization problems, while MHs are general algorithms applicable to a wide range of problems. MHs operate by repeated objective function evaluations without using gradient information (Gradient-Free), aiming to find a global optimum. Although they don't guarantee optimal performance, they are effective in solving complex problems, particularly combinatorial optimization problems. MHs can be memoryless, like Genetic Algorithms (GA) and Simulated Annealing (SA), or they can use memory, as in Tabu Search (TS), to avoid repetitive decisions. Additionally, MHs can be deterministic (e.g., TS) or stochastic (e.g., GA, SA). MHs are also categorized as single-solution-based (e.g., TS, SA) or population-based (e.g., GA, Particle Swarm Optimization), depending on whether they evolve a single solution or a population of solutions.

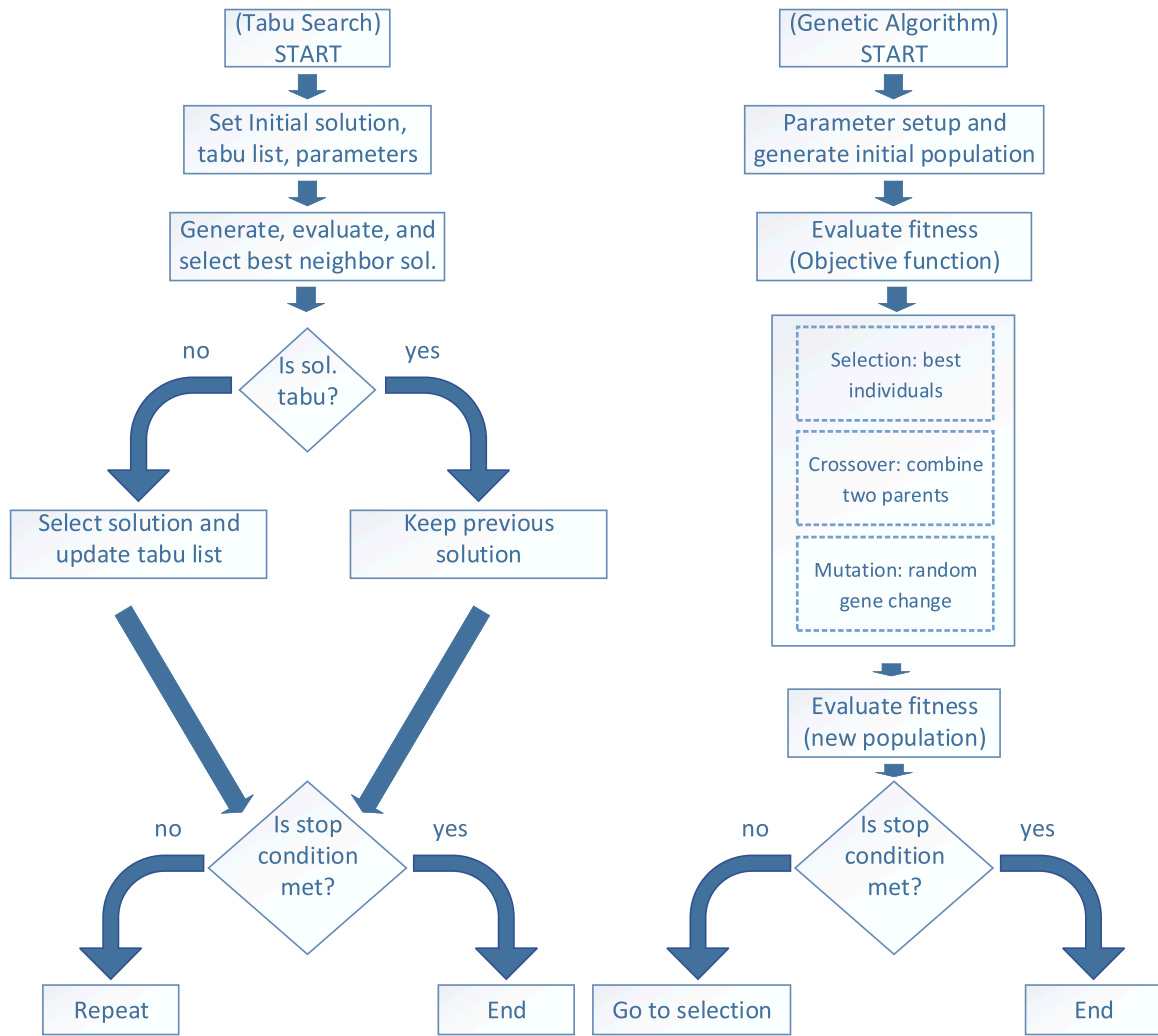


Fig. 2. Typical structure for Tabu search and genetic algorithm.

1.2.3. Nominated Tabu search and genetic algorithm as evaluating benchmarks

Tabu Search, introduced by Fred W. Glover, is a metaheuristic known for strong local search and quick convergence. Unlike traditional methods, such as steepest descent, TS accepts non-improving solutions to escape local optima and uses a tabu list to prevent revisiting previous solutions, promoting exploration. This makes TS effective for large-scale combinatorial optimization (Karamichailidou et al., 2021), where the goal is to identify the best solution from a finite set of possibilities. For example, (Hafeez et al., 2021) applied TS to optimize decision tree parameters, framing it as a discrete optimization task aimed at minimizing errors and improving accuracy.

Genetic Algorithm, falling under the umbrella of evolutionary algorithms, is a more advanced and popular MH optimization method. It optimizes through selection, crossover, and mutation. In the field of

digital hydraulics, GA has been applied to optimize solenoid actuator response times while satisfying energy and size constraints (Pellikka et al., 2011). At Tampere University, GA was further utilized to improve energy efficiency in multi-pressure hydraulic systems by identifying optimal control trajectories that minimized steady-state throttling and switching losses during mode transitions (Huova & Linjama, 2022). However, some researchers have noted that while GAs are powerful for real-world problems, they can struggle with large solution spaces and numerous local optima, and may not reliably find the global optimum (D'Angelo & Palmieri, 2021). (Fig. 2)

1.2.4. Exploiting reinforcement learning for solving optimization problems

Unlike MHs methods, which are mainly designed for static optimization problems and often require significant computational effort to adapt to dynamic changes, Machine Learning (ML) dynamically learns

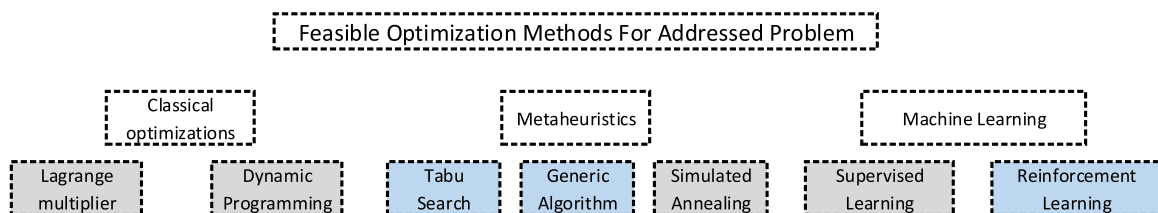


Fig. 3. Selected collection of optimization methods for the addressed problem, with nominated algorithms highlighted in blue.

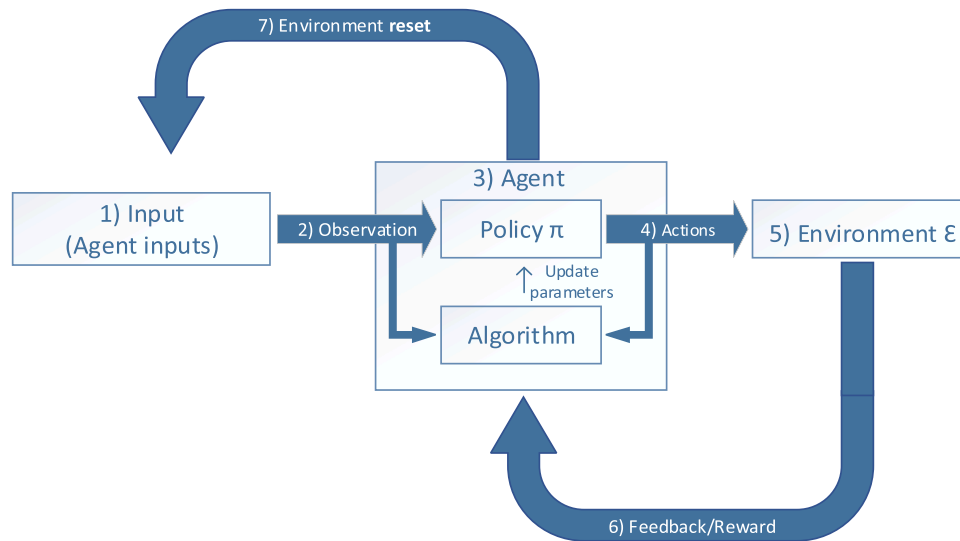


Fig. 4. Sequential decision making: single step reinforcement learning cycle with environment reset.

optimal actions through feedback, making it better suited for problems where system conditions or objectives evolve continuously. Optimization algorithms seek to maximize or minimize an objective function, while ML involves developing algorithms that improve performance through experience and adapt to new tasks. Recently, integrating ML techniques with MHs has gained significant attention (Bengio et al., 2021).

A key area of ML relevant to optimization is Reinforcement Learning (RL), where an agent learns to make decisions by interacting with its environment and maximizing feedback rewards. RL algorithms typically operate in two phases: (1) the training phase, where the distinction between policy being learned is the *same* as the policy used to generate data (on-policy) or *different* (off-policy) is relevant, and (2) the inference phase, where the learned policy—whether value-based or policy-based—is used to make decisions in the environment.

Neural networks, particularly within reinforcement learning (RL), have been widely used for combinatorial optimization (Garmendia et al., 2022). Unlike supervised learning, which relies on large labeled datasets requiring domain expertise, RL methods derive knowledge directly from interaction, eliminating the need for labeled data and enhancing adaptability to new problems (Wang et al., 2024). (Fig. 3)

1.2.5. Single Step (No Transition) RL Approach

Deep Reinforcement Learning (DRL) has improved complex decision-making, especially in state-independent (single-step) scenarios, though its full effectiveness is still under evaluation (Viquerat et al., 2023). Standard RL is inefficient for single-step tasks, requiring precision adjustments. One study tackled this inefficiency by using an improved TD3 for better sample efficiency (Luo et al., 2023), applying a single action per episode with environment resets to its initial state, similar to our problem setup.

Viquerat's team (Renault et al., 2023) proposed single-step Proximal Policy Optimization (PPO-1) for state-independent gas furnace control and later introduced Policy-Based Optimization (PBO) (Viquerat et al., 2023), a single-step DRL algorithm inspired by evolution strategies to enhance efficiency. In another example, (Fricke et al., 2023) applied RL to optimize flow channel shapes in extrusion dies, aiming for one-step geometry optimization per episode, though achieving it in one episode was not mandatory. Additionally, (Kharrat et al., 2024) introduced a stateless actor-critic RL approach within a continuum bandit framework, designed for continuous-action optimization, emphasizing efficient prompt generation without relying on state transitions or future rewards. Although Policy-gradient algorithms such as PPO-1 and PBO can

be applied to discrete action spaces, our optimization problem is intrinsically a single-step contextual bandit. Adopting a bandit formulation enables straightforward value updates and avoids the high variance in gradient estimates and the intensive hyperparameter tuning typically associated with policy-gradient methods.

In the contextual multi-armed bandit field, related work includes Contextual Multi-Armed Bandit with Deep Reward Prediction (Stylianopoulos et al., 2022), Contextual Multi Arm Bandit with Deep Q-Network (Z. Li & Ai, 2023) and Neural Network-based Bandit approaches (Raghuwanshi et al., 2024). Thompson-sampling variants such as the neural-linear bandit choose actions by sampling from a Bayesian posterior and then updating a full covariance matrix—an operation the authors flag as the chief computational bottleneck (Russo et al., 2018). To avoid this cubic-time burden, this article adopts a lighter ϵ -greedy deep contextual-bandit algorithm that achieves modest accuracy with faster learning and far smaller memory requirements. (Fig. 4)

1.3. Objective, novelty, limitations and structure

Purpose and framework

This study aims to develop an interaction-based optimization method to minimize flow fluctuations during transient states in Digital Hydraulic systems, with a strong potential for real-world implementation. To the best of the authors' knowledge, this is the first study to apply a Deep Contextual Bandit (DCB) framework to digital flow regulators incorporating both direct and pilot-operated valves or to mitigate flow fluctuations. The approach is specifically tailored for the largest Digital Flow Control Unit ever presented, which escalates dramatically the issue of fluctuations.

Readers should differentiate between three cases: (1) multi-step episodes without resets, including those with single-step lookahead (Mnih et al., 2015), which are not our focus; (2) single-instance learning, where decisions are sequential and impact the same problem instance—out of scope (Bengio et al., 2021); (3) single-step, multi-episode learning (in scope), where inputs are independent, with no state transitions between episodes.

Limitations

Several limitations are acknowledged. *First*, the differential pressure across the valves is capped at 20 bar to avoid exponential flow increases. *Second*, flow variation is constrained by the physical dynamics of the valves and the controller response time. *Third*, although temperature is known to influence hydraulic fluid viscosity and valve dynamics, it is not

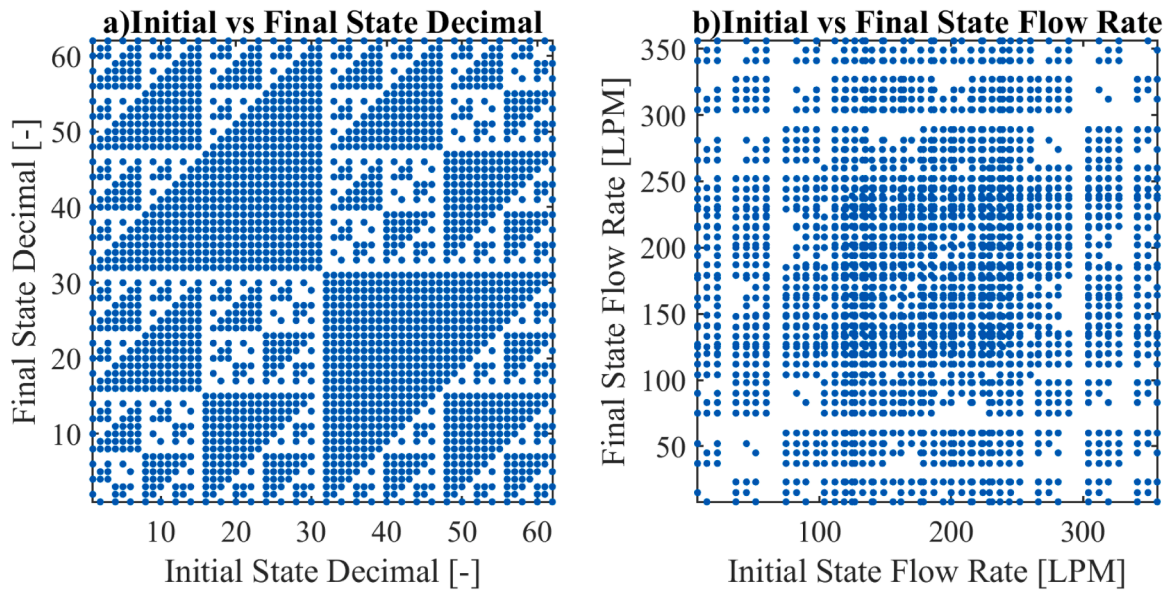


Fig. 5. Transition maps between valve states in a 6-Valve DFCU. (A) Ideal configuration using uniform binary steps; (B) Actual configuration based on semi-binary valve sizing, mapped to physical flow rates [LPM]. Each point represents a valid transition between initial and final flow states.

considered in this study, however its implementation methodology was briefly discussed. *Fourth*, the current policy assumes nominal valve response characteristics. If ageing causes the valve to respond more slowly over time, the pre-trained delay policy may become suboptimal, as no online adaptation or drift-tracking mechanism is included. These exclusions are intentional to isolate and focus on the dominant variables affecting flow synchronization. *Finally*, this study uses streamlined optimization to address system limits without complex algorithms variations or hardware changes.

Outline

Given the insights from Sections 1.1 and 1.2, the following steps are taken to address the synchronisation issue:

- **Section 2:** Problem Setup – System description and objective definition.

Table 1

Specifications of valves used in the DFCU ($n = 6$).

Main Valves	Nominal Flow LPM @ Δp 5bar	Operation	Orifice / Stroke	Manufacturer
DV1	8	Direct Operated poppet	2.8 mm orifice.	Enhanced WS22GD-**
DV2	15	Direct Operated poppet	4 mm orifice.	Bucher Hydraulics
DV3	37	Direct Operated poppet	none	
POV4	75	Pilot Operated NG16 actuated by the Enhanced pilot valves WS22GD: In-PV4, and Out-PV4	3.1 mm stroke.	Rexroth LC valve with LFA-HWM cover.
POV5	104	Pilot Operated NG16 actuated by the Enhanced pilot valves WS22GD: In-PV5, and Out-PV5	4.3 mm stroke.	
POV6	125	Pilot Operated NG16 actuated by the Enhanced pilot valves WS22GD: In-PV6, and Out-PV6	5.2 mm stroke.	

- **Section 3:** Discrete Optimization Implementation – Highlights Tabu Search and Genetic Algorithm.
- **Section 4:** Proposed Approach – Description of the developed RL method.
- **Section 5:** Evaluation – Performance testing and comparison with benchmarks.

2. Mathematical modeling and peaks explaining

2.1. Proposed system configuration

2.1.1. General design criteria

The proposed DFCU is designed to match the flow–pressure performance ($Q-\Delta p$) characteristics of NG25 two-way, two-position pilot-operated proportional valves. Comparable devices include the Parker TDA, Rexroth 2WFC, and Vickers CVU-**-EFP1 valves, which typically offer flow capacities of approximately 350 LPM at $\Delta p = 5 \text{ bar}$. High performance valves such as the Parker TFP can reach up to 650 LPM (NG25). However, they require significant pilot pressure, where the proposed DFCU require minimal control pressure. Given this analysis, the DFCU targets an average flow rate of $Q_{target} = 500 \text{ LPM}$, forming the *First* design constraint. The *Second* constraint focuses on matching the precision control of high-end pilot-operated solenoid proportional valves, with an error band ϵ of 2% to 4%. In a binary-coded DFCU, the error is calculated as the ratio of the largest difference between two successive flow steps to the maximum flow. The largest difference between two successive flow steps in an ideal DFCU is the base flow unit q_0 .

A search grid is constructed to explore different configurations of binary valves n and base flow rates $q_0(n) = \frac{Q_{target}}{2^n - 1}$. Calculating the approximate total flow $Q_{approx}(n) = q_0(n) \cdot (2^n - 1)$ and their inaccuracies error $(n) = \frac{q_0}{Q_{approx}}$ for each setup. The optimal solution $n_{optimal}$ involved 6 binary valves, with a base flow rate of $q_{0,optimal} \approx 8 \text{ LPM}$ yielding a total flow of $Q_{approx} = 504 \text{ LPM}$ and an error margin of $\epsilon = 1.6\%$, well within the acceptable range. Due to system constraints, a semi-binary valve configuration was adopted, with valve ratios $V1 : V2 : V3 : V4 : V5 : V6 = 1 : 1.9 : 4.6 : 9.4 : 13 : 16$ (Although not fully implemented, it is recommended that each valve be nearly less than twice the size of the previous one (Laamanen et al., 2007)). This setup achieves a flow rate of 363 LPM with a 4% error, meeting the upper limit. Unlike the ideal

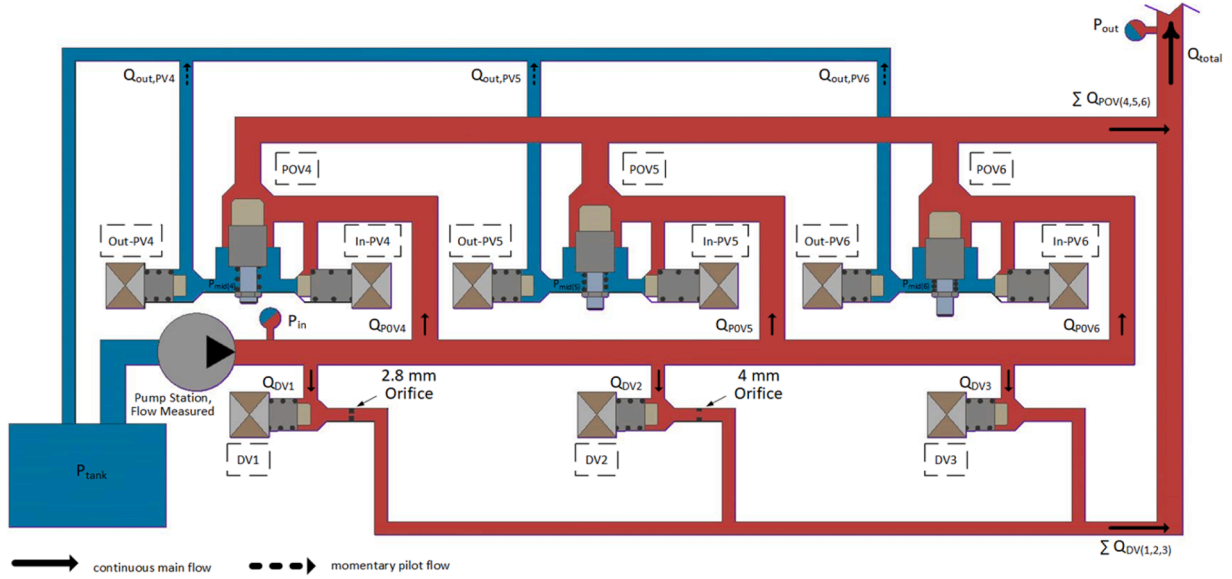


Fig. 6. DFCU($n = 6$) Layout within a simplified test environment; All main valves shown in open state.

PCM system with a doubling ratio of $1 : 2 : 4 : 8 : 16 : 32$ and a theoretical resolution of $63[-]$, this configuration balances practical limitations while striving to meet performance goals.

2.1.2. System representation

The binary states of the valves and the corresponding flow rates can be described as follows: Each configuration represents a unique combination of flow settings across six valves, resulting in $2^n = 64$ different flow “states” (including zero), and each state can transition to $2^n - 1$ (excluding itself) other state, yielding $(2^n) \times (2^n - 1) = 4032$ flow shifts as illustrated in Fig. 5. However, exactly $\frac{2702}{2}$ of these transitions cause pressure peaks in a DFCU ($n = 6$). This number was calculated by identifying cases where at least two valves switch states in opposite directions, leading to either a temporary flow increase (*crest*) or decrease (*trough*), similar to wave patterns.

The operational characteristics of the selected valves are key to understanding the system response. Direct-operated valves (DVs), and similarly Pilot valves (PVs) exhibit switching times of 21 ms (open) and 14 ms (close), based on the experimental results of (Ketonen & Linjama, 2017). Pilot-operated valves (POVs) show a slight delay of less than 5ms, as observed in simulation studies (Elsaed & Linjama, 2024a). The specifics of the valves are detailed in Table 1.

Both the pilot valves (PVs) and the direct valves (DVs) use the exact

by two pilot valves (PVs), with dynamics captured by $u_{in,act,PV(j)}$ and $u_{out,act,PV(j)}$.

$$Q_{total}(t) = \sum_{idv=1}^3 Q_{DV(idv)}(t) + \sum_{ipov=4}^6 Q_{POV(ipov)}(t) \quad (3)$$

$$Q_{DV(idv)}(t) = u_{DV(idv)}(t) \cdot k_{v,eff(idv)} \cdot \sqrt{p_{in}(t) - p_{out}(t)}$$

Here, $\sqrt{\Delta p}$ is $sign(\Delta p) \cdot \sqrt{|\Delta p|}$, with $idv = 1, 2, 3$ corresponds to DV1, DV2, and DV3 valves. $p_{in}(t)$, $p_{out}(t)$ are inlet and outlet pressures.

Direct-operated valve models were simplified by omitting minor parameters like pressure build-up, as these have minimal impact on flow control accuracy. The valve response time remains consistent across pressures, as confirmed experimentally in (Ketonen & Linjama, 2017).

$$Q_{POV(ipov)}(t) = C_d \cdot A_{out,POV(ipov)}(t) \cdot \sqrt{2 \times (p_{in}(t) - p_{out}(t)) / \rho}$$

Where, ρ and C_d are constants, while $A_{out,POV(ipov)}(t)$ is variable and defined as:

$$A_{out,POV(ipov)}(t) = x_{m,POV(ipov)}(t) \times \sin(\alpha) \times \pi \times D_{out}$$

Here, α and D_{out} are constants, and $ipov$ corresponds to pilot-operated valves 4,5 and 6. The motion $x_{m,max,(ipov)}(t)$ is determined by the equation:

$$m_m \ddot{x}_{m,POV(ipov)}(t) = -c_m \dot{x}_{m,POV(ipov)}(t) - k_m (x_{m,POV(ipov)}(t) + x_{preload}) + (p_{in}(t) \cdot A_{st} + p_{out}(t) \cdot A_{out,POV(ipov)}(t) - p_{mid}(t) \cdot A_{mid})$$

valve model; the only difference is DV1 and DV2 have attached orifices.

The test setup, including the DFCU layout and environment, is presented in Fig. 6.

2.2. Mathematical writing for equations

2.2.1. System interdependencies and dynamic modeling

Total flow in the Digital Hydraulic system is governed by a combined flow equation. Flow is computed through two valve sets: (1) DVs, using a simplified orifice equation with dynamics captured by $u_{DV(i)}(t)$, and (2) POVs, with more complex dynamics, including main poppet movement and pressure build-up in the poppet chamber. These POVs are controlled

Where, m_m , c_m , k_m , $x_{preload}$, A_{st} , A_{mid} are constants. $p_{mid}(t)$ can be solved using the following equation.

$$\frac{dp_{mid,POV(ipov)}(t)}{dt} = \frac{\beta}{V_{mid,POV(ipov)}(t)} (Q_{in,PV(ipv)}(t) - Q_{out,PV(ipv)}(t) - \dot{V}_{mid,POV(ipov)}(t))$$

Where, β is a constant. While $V_{mid,POV(ipov)}(t)$ is given by: $\frac{dV_{mid,POV(ipov)}(t)}{dt} = -A_{mid} \cdot \dot{x}_{m,POV(ipov)}(t)$

On the other side, the inlet flow is defined as :

$$Q_{in,PV(ipv)}(t) = u_{in,act,PV(ipv)}(t) \cdot k_{v,PV} \cdot \sqrt{p_{in}(t) - p_{mid(ipov)}(t)}$$

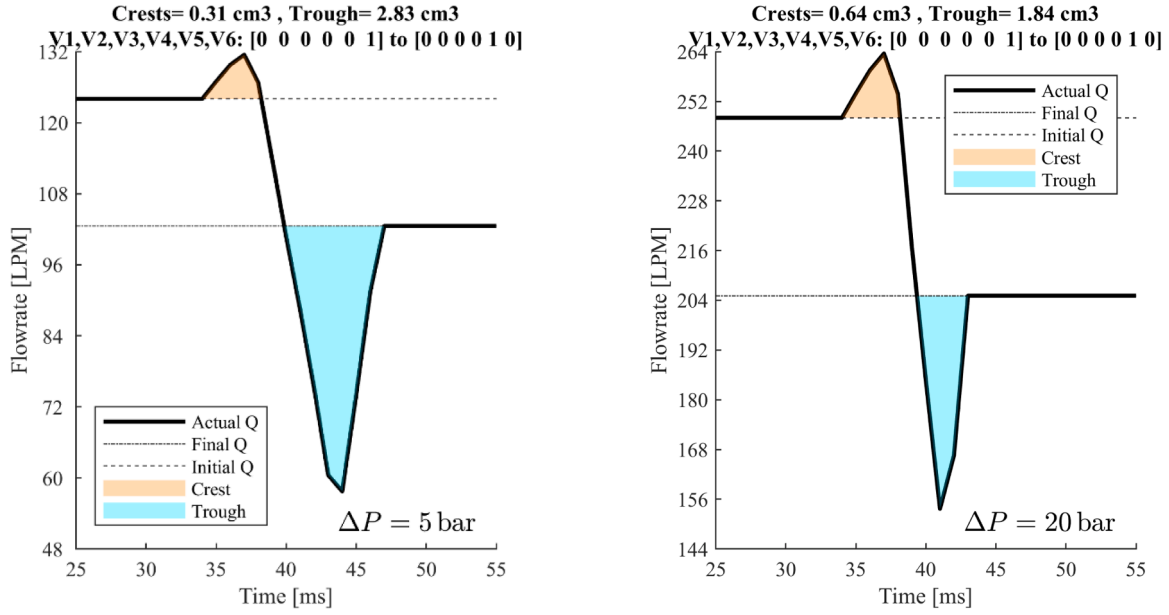


Fig. 7. Effect of pressure level on flow-rate crests and troughs; higher Δp shortens peak duration.

Similarly, the outlet flow is:

$$Q_{out,PV(ipv)}(t) = u_{out,act,PV(ipv)}(t) \cdot k_{v,PV} \cdot \sqrt{p_{mid(ipov)}(t) - p_{out}(t)}$$

Where $k_{v,PV}$ is constant, and $ipv = 1, 2, 3$. It is evident that $A_{out,POV(ipov)}(t)$ and $x_{m,(ipov)}(t)$ and $p_{mid}(t)$ are closely interdependent.

Valves Inherent Response: In addition to the artificial delays, which are introduced and determined by the optimizer, the valves also exhibit inherent delays. These inherent delays are modeled to the binary input signals (0 or 1), where $u_{input,k}(t)$ is a constant input for different valves. Here, k represents directional valves $DV(1)$, $DV(2)$, $DV(3)$ or pilot valves ($in, act, PV(4)$, $in, act, PV(5)$, $in, act, PV(6)$, $out, act, PV(4)$, $out, act, PV(5)$, $out, act, PV(6)$). A transport delay and rate limiter model hydraulic dynamics, simulating inertia, damping, and flow forces (Luo, 2006). The transport delay reflects the system inherent response time, while the rate limiter controls valve movement speed. This method balances computational efficiency with accuracy, optimizing delays for electrically operated valves (three DVs and six PVs) while POVs respond accordingly.

The transport delay $d(t)$ shifts the input by $t_d = 9ms$ (commonly taken as 2/3 of the expected total response time) resulting in $u_{delay,k}(t) = u_{input,k}(t - t_d)$. Then a rate limiter constrains the signal rate to a $t_{rise} = 13.5ms$ and $t_{fall} = 5.3ms$ yields $t_{open/close} = t_d + t_{rise/fall}$. A 0.1ms maximum step size for the ode23tb solver balances accuracy and computational resource.

2.2.2. Target and simulink optimization platform

Optimization objective. Given the system complex, nonlinear response during transitions, this study applies AI-based RL for black-box optimization, with TS and GA as benchmarks (Kumagai & Yasuda, 2023).

Delay Injection Model: The optimizer adjusts an artificial delay $t_{Ar,k}$ for each valve k to better align the timing between faster closing and slower opening valves. The input signal is modified as:

$$u_{delay,k}(t) = u_{input,k}(t - t_d - t_{Ar,k}). \quad (4)$$

The algorithm explores delay configurations for up to eight switching valves, selected from a pool of nine electric-operated valves. Since at least two valves must switch in opposite directions for a peak (crest or trough) to occur, this means one valve will always shift in the opposite

direction (e.g., one valve is opening when the other eight valves are closing). The configuration distribution for the 2702 cases is categorized as follows: 1 delay ($n_d = 1$) occur in 633 cases, 2 delays: 828 cases, 3 delays: 604 cases, 4 delays: 366 cases, 5 delays: 186 cases, 6 delays: 64 cases, 7 delays: 18 cases, 8 delays: 3 cases. The total iterations required to fully explore the DFCU can be expressed as.

$$Total\ Iterations = T_{readings} \times P_{readings}$$

$$\times \sum_{i=1}^{n_d=8} (number\ of\ cases\ for\ valve\ i \times (OC \times D)^i) \quad (5)$$

Brute Search: To address the synchronization problem, delays can be applied to both opening and closing events ($OC = 2$), considering 2 temperature readings, 3 pressure levels, and 4 delay values, a brute-force approach may require *680 million* iterations, taking nearly one year! at 50 ms per iteration.

Brute Simplification: The complexity increases significantly when considering both opening and closing delays. By setting $OC = 1$ (focusing only on closing delays) and fixing the temperature (moreover under fully turbulent flow, C_d drift is negligible), the iterations reduce to *1.1 million*. Notably, delays for valves 6, 7, and 8, which represent just 3% of the 2702 cases, account for 70% of the iterations. The size and complexity of the search space make brute-force optimization impractical. This highlights the need for AI to improve efficiency and adaptability. Given the computational demand, such problems are typically solved numerically rather than analytically on standard PCs.

It should be noted that, the optimization task is not to select switching scenarios or control which valve activates, these are fixed by the digital controller explained in (Elsaed & Linjama, 2024a). The optimizer role here is to learn the optimal timing (delays) for executing these predefined cases.

Simulink optimization platform.

Step 1. Define Action Search Space: Notably, crests and troughs persist longest at the lowest pressure trained ($\Delta p = 5$ bar), which also represents a typical operating point for most industrial circuits. Consequently, the delay pool, independently $D = \{0, 2, 4, 8\} ms$ is optimised at this 5-bar condition and the same candidate set is reused for other Δp values. It should be mentioned that only the *menu* of

delays is fixed; the optimizer still chooses a fresh value for each pressure case.

Step 2. Define State and Inputs: The environment state includes initial and final flow conditions and the current pressure differential ΔP . These variables are encoded in a single-pulse signal of length $t_{sim} = t_{ini} + 30ms$. The 30 ms window comfortably covers the full crest-and-trough transient, yet keeps each simulation lightweight for RL training. The digital hydraulic controller extracts these states to generate a valve command vector U . This methodology, previously established in literature, is not detailed here.

Step 3. Simulate Environment and Evaluate Performance Metric: After running Simulink Model, a script detailed in (Elsaed & Linjama, 2024a) noisily analyze major spikes (crests/troughs), filtered minor deviations, and identify the error volume cm^3 , simulating real-world empirical measurement. (Fig. 7)

3. Impelmenting discrete optimizers

3.1. Tabu Search implementation

Tabu Search (TS) is used to optimise artificial delays across up to eight switching valves in one loop. The same structure is applied in the GA baseline for comparison. A fitness function simulates each configuration and evaluates the resulting flow error, with a performance threshold of $\epsilon = 0.2cm^3$.

TS Algorithm: Tabu Search for Valve Delay Optimization

1. Initialization: (for each case index $Case_{ic} = 1$ to 2702):

- Set initial solution s^0 with zero delays.
- Determine the active valve count n_d for the current case.
- Evaluate the initial solution error using the fitness function, $f(s^0)$.
- Set the best solution $s_{best} = s^0$ and $f_{best} = f(s^0)$.
- Initialize the Tabu List: $TabuList = \{\}$.
- 2. Input:
 - Valid delay times: $D = \{0, 2, 4, 8\}ms$.
 - Error threshold: $\epsilon = 0.2cm^3$.
 - Set limits: $MaxIter, MaxAttempts, NumNeighbors: \max([0.1 \times (3^n - 1)], 20)$
- 3. Main Optimization Loop (for each $iter \leq MaxIter$):
 - Generate neighboring solutions $N(s^{iter-1})$, modify delays excluding moves in $TabuList$ and evaluate them $f(s)$.
 - If $f(s^{iter}) < f_{best}$ update best overall solution $s_{best} = s^{iter}$ and error $f_{best} = f(s^{iter})$.
 - Add new neighbors to Tabu List.
 - Store s_{best} and f_{best} for Case ic.
 - Stop if criteria are met.
- 4. End of Loop Over All Cases.
- Repeat for all cases.
- 5. Output:
 - Return best solutions $\{s_{best}^{(ic)}\}$ and errors $\{f_{best}^{(ic)}\}$ for all cases.

The algorithm begins with a zero-delay solution to minimize response time and avoid unnecessary delays. In each iteration, delays values are adjusted to either increase or stay the same within the boundaries for each variable, with the number of neighbors limited by $NumNeighbors$ to reduce computational load. The $TabuList$ tracks tried solutions to avoid repeats, while $maxAttempts$ prevents endless loops when no valid neighbors are available. Using a Tabu List with long-term memory boosts performance by avoiding local minima, in contrast to traditional Tabu Search with short-term memory (Chen et al., 2024).

3.2. Genetic algorithm implementation

A genetic algorithm is used to optimize artificial delay values across multiple valves, minimizing flow peaks during transitions. Through grid search-based tuning of key hyperparameters, this nature-inspired al-

Table 2

Global optimization of valve delays with combined actions, where $A=D^i$, at $D = 4$.

Switching Valves count	1	2	3	4	5	6	7	8
Action space size	4	16	64	256	1024	4096	16,384	65,536

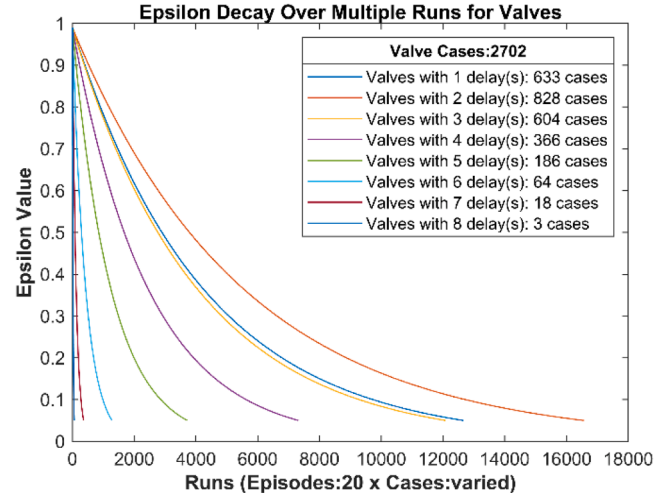


Fig. 8. Epsilon decay over runs for each valve configuration.

gorithm efficiently navigates the discrete delay space of the system.

GA Algorithm: Genetic Algorithm for Valve Delay Optimization

1. Initialization: (for each case index $Case_{ic} = 1$ to 2702):

- Set initial solution s^0 with zero delays.
- Determine the active valve count n_d .
- 2. Input:
 - Valid delays: $D = \{0, 2, 4, 8\}ms$.
 - Error threshold: $\epsilon = 0.2cm^3$.
 - Set Population Size $N_p = \min(\max([0.1 \times (3^n - 1)], 15), 25)$, Maximum Generations $G_{max} = \min(\max([0.1 \times (3^n - 1)], 10), 15)$, Mutation Rate $\mu = 0.05$.
- 3. Main Optimization Loop (for $iter \leq G_{max}$):
 - Selection: Sort population P based on their errors (fitness value) $f(P_i)$ and select top half $P_{Parents}$, where $i = 1, 2, \dots, N_p$.
 - Crossover: Randomly pair parents from $P_{Parents}$ to generate new offspring O via recombination: $O = [P_{Parent1}(1:c), P_{Parent2}(c+1:n_d)]$.
 - Mutation: Apply random delay changes to offspring O_i where $O_i(j)$ is Random choice from D with probability μ , where $i = 1, 2, \dots, N_o$ (number of offspring) and $j = 1, 2, \dots, n_d$.
 - Evaluation: For each offspring compute fitness $f(O_i)$ and update s_{best} and f_{best} if an improved solution is found.
 - Stop if $f_{best} \leq \epsilon$ or constraints are met.
- 4. End of Loop Over All Cases.
- Repeat for all cases.
- 5. Output:
 - Return best solutions $\{s_{best}^{(ic)}\}$ and errors $\{f_{best}^{(ic)}\}$ for all cases.

Besides the drawback discussed in the introduction, another limitation of the standard GA implementation is its fixed parameters, limiting dynamic balance between exploration and exploitation/commit. This leads to suboptimal performance, as it cannot adjust mutation and crossover rates based on search progress. For instance, mutation rates should decrease once a good solution is found and increase to escape local optima to promote exploration (Karimi-Mamaghan et al., 2022).

4. Reinforcement learning realization: deep contextual multi armed bandit (DCB)

The problem resembles coordinating an orchestra, where a single conductor (agent) determines the timing (action) of each musician (valve), each playing only a single strike of an instrument (valve closing). Each musician can have one unique timing (delays: {0, 2, 4, 8}) milliseconds. These adjustments ensure the single note (environment) sounds smooth (reduce flow peaks) across discrete independent notes (2702 switching scenarios), and for varying continuous sounds levels (differential pressures). The relationship between timing and smoothness (reward) is uncertain.

4.1. Sequential strategies: single step formulation

Unlike full Reinforcement Learning approaches, which rely on Markov Decision Processes and Bellman equations to model and solve multi-step decision-making problems, the contextual bandit (CB) framework simplifies the setting to a single step per episode, focusing solely on immediate rewards. Nevertheless, CB still involves sequential learning, as the agent improves its policy based on past interactions feedback. The presented problem aligns with a standard contextual bandit given the system selects a single action, and then only a single aggregate reward is received without observing the rewards of other possible actions (Y. Li et al., 2024).

Each switching scenario is solved in one shot: the agent chooses the entire delay vector, the simulator runs to completion, returns a single reward, and resets. No intermediate observation is provided, and the next scenario initial state is fixed by the scenario set, not by the previous action. Therefore, the task instantiates the single-state limit of RL—also known as a contextual bandit (Slivkins, 2019). Contextual Bandits, based on Robbins multi-armed bandit problem (Robbins, 1952) and developed further by researchers, extend the classic model and excel in adaptive online learning, adjusting strategies to counter long-term performance degradation in hydraulic components.

4.2. Algorithm to solve optimal policies in RL

The action-value function $\hat{q}(x, a)$ is parameterized to model the relationship between the observed context and the immediate feedback (Mazyavkina et al., 2021). In a simplified DFCU scenario, the system observes a context x (current: $Q_{initial}$, Q_{final} , ΔP) and selects a combinatorial action a (a set of delay configurations for multiple valves, where each action represents an arm), and receives immediate feedback r (based on the resulting deviated flow volume in cm^3). Since this is a contextual multi-armed bandit problem with a horizon of $H = 1$, the value function is $\hat{q}(x, a) = E[R_t | X_t = x, A_t = a]$.

The used method, while presented here in a single-step contextual bandit framework, remains effective for a broad class of RL-Combinatorial problems (Mazyavkina et al., 2021), particularly those characterized by discrete action spaces. Even when dealing with low-dimensional inputs, the complexity arising from the input-output mapping justifies employing a deep neural network. By leveraging a DNN, the approach can capture intricate relationships between contexts and actions.

During training, parameterized action-values are refined incrementally, adjusting estimates to reduce discrepancies between observed and predicted immediate rewards. In this single-step contextual setting, the update simplifies to: $\hat{q}(x, a; \theta) \leftarrow \hat{q}(x, a; \theta) + \alpha(\delta_i)$, with prediction error $\delta_i = R(x, a) - \hat{q}(x, a; \theta)$ where α is the learning rate. By applying this adjustment repeatedly, the model converges to accurate estimates of the expected reward for each action, enabling selection of actions that yield higher immediate returns. The network takes the given context as input and produces parameterized action-values for all feasible actions simultaneously, facilitating efficient evaluation. These parameters θ are updated during training to minimize the loss function: $L(\theta) = \sum_{j=1}^M \delta_j^2$, where M is the mini batch size. The δ compares the predicted action-value with the actual reward received. This incremental process effectively handles noise or stochasticity in rewards, steadily improving estimates while maintaining stability. Techniques like replay buffers and mini-batch sampling further enhance sample efficiency, making this method particularly suitable for single-step decision tasks in complex discrete action spaces.

The Adam optimizer trains the parameterized action-value network efficiently with low memory use. While leveraging NN for approximating complex input-output mappings is beneficial, it risks overfitting with limited environmental interactions. To mitigate this, L2 regularization is applied, with a weight decay λ range of 1×10^{-8} to 5×10^{-7} , as demonstrated to be effective in a supervised learning study on a simple digital hydraulic control unit (Elsaed & Linjama, 2024d). The NN employs ReLU activations ($\text{ReLU}(x) = \max(0, x)$) in the hidden layers, which is commonly effective for this type of observations (Coskun & Itik, 2024), processing input context to parameterized action-value for each action:

$$\hat{q}(x, a; \theta) = W_{out} \cdot \text{ReLU}(W_3 \cdot \text{ReLU}(W_2 \cdot \text{ReLU}(W_1 \cdot x + b_1) + b_2) + b_3) + b_{out} \quad (6)$$

The output layer of the NN contains neurons equal to the total number of possible actions (Table 2), which can reach thousands for higher-valve models but occurs in fewer cases. By producing parameterized action-values for all actions in a single forward pass using only the context, thereby avoiding the need for multiple forward passes required when both context and action are inputs—a method that often requires additional techniques to make the process feasible (Fourati et al., 2024). Although the used approach increases memory overhead (which remains manageable), it is generally regarded as advantageous for reducing inference time in discrete constrained action spaces.

4.3. Actions approach and exploration strategy

In the presented DFCU, multiple actions affect the environment simultaneously. Three distinct approaches will be evaluated that define the action space A as all possible combinations of delay values $D = \{d_1, d_2, \dots, d_n\}$ assigned to i valves, resulting:

- Independent Actions per Valve: The agent selects independent delay values from D , treating each valve separately; this enhances scalability but possibly missing global optima.
- Multi-Agent System: Each valve has an individual agent, enabling coordinated, dynamic interactions but requiring more computational resources (Hu et al., 2024).
- Combined Actions (Table 2): Agent selects from D^i combinations, considering the global effect of all valves simultaneously, optimizing globally but facing exponential complexity with larger i . Therefore, this approach will be selected offering a balance for systems with a manageable number of valves.

Based on the selected method, the problem becomes not a combinatorial bandit in the traditional sense, as the action space consists of predefined atomic actions (combinations of valve delays), and the agent selects directly from this set without combining components (arms) during decision-making.

Common strategies for exploitation-exploration balance in CB problems include ϵ -greedy, upper confidence bound, and Thompson sampling. For this application, ϵ -greedy is chosen due to its simplicity and adaptability to stochastic environments, where the agent either exploits the action with the highest parameterized action-value or explores randomly with probability ϵ . To provide the reader with valuable insights, it is worth noting that recent methods like (Ban et al., 2022) use more sophisticated techniques such as an Exploration Neural Network, to adaptively target arms with higher potential while reducing exploration of low-potential arms.

$$\text{Action } a = \begin{cases} \text{random action from } A, \text{ with probability } \epsilon \\ \underset{a}{\operatorname{argmax}} \hat{q}(x, a, \theta), \text{ with probability } 1 - \epsilon \end{cases} \quad (7)$$

The DCB with ϵ -greedy approach allowing dynamic ϵ decay—starting with high exploration that decreases as more episodes are completed. The decay of ϵ is controlled by: $\epsilon_{\text{decay}} = 1 - \left(\frac{\epsilon_{\min}}{\epsilon_0}\right)^{\frac{1}{N_{\text{episodes}} \cdot N_{\text{cases}}}}$, where $\epsilon_0 = 0.99$, $\epsilon_{\min} = 0.05$, $N_{\text{episodes}} = 20$ and N_{cases} is the number of cases. At each iteration, the exploration rate updates as: $\epsilon_{i+1} = \epsilon_i \times (1 - \epsilon_{\text{decay}})$, ensuring that ϵ_{\min} reaches 0.05 at the end. (Fig. 8)

The reward function guides the agent toward optimal solutions by assigning a reward of 8 for zero or near-zero measured volume deviation and $R = -\log(\text{Vol}_{\text{cm}3})$ for others. The immediate reward is received after the selected action, based on the specific context, influences the environment and the resulting volume deviation is measured. To emulate real-world conditions, Gaussian noise $N(R, 0.5)$ was added to the reward signal, introducing uncertainty into the environment. This noise magnitude reflects typical variability observed in typical physical systems. Stochastic environments incorporating Gaussian noise have been previously implemented in the CB-based NN, particularly under UCB-based exploration strategies (Zhou et al., 2020).

4.4. Script for full interaction set (2702) at single ΔP

Hyperparameters were tuned: A sample-time aligned with event-driven actions was adopted; a learning-rate of 0.03 gave fast, stable convergence, and a mini-batch of 16 preserved adaptability when exploration was sparse. The final setup used 20 training episodes, a replay buffer twice that length, and a 3-layer MLP (256–128–64).

Hyperparameters Sensitivity Analysis: Selecting a suitable network

architecture is pivotal for fair baselines and underscores the promise of more efficient and robust hyper parameters (Henderson et al., 2018). Accordingly, a one-factor sweep was carried out to characterise sensitivity: learning-rate {0.01, 0.03, 0.10}; mini-batch size 8 – 32; weight-decay $0 - 1 \times 10^{-4}$; episode budget 10 – 40; replay-buffer length from twice the episode count up to 10,000; and three MLP widths (64–32, 256–128–64, 1000–1000) under ReLU activations. Only weight decay, max episodes and NN architecture dominate performance, learning-rate and mini-batch size slightly tune the final error, while the remaining dimensions altered performance smoothly and marginally.

DCB Algorithm: Agent Training for Valve Delay Optimization

1. Initialization:

- Define 8 different active valve configurations ($n_d = 1$ to $n_d = 8$).
- 2. Main Loop: Over the active valves (for $n_d \leq 8$):

- Identify cases $\{Case_{ic}\}_{n_d}$ corresponding to each n_d .
- Load system parameters and initial and final states.
- User-Defined Parameters:
 - Error threshold: $\epsilon = 0.2\text{cm}^3$.
 - Training parameters.
- a) Inner Training Loop for Each $\{Case_{ic}\}_{n_d}$:
 - Define the action space $A_{n_d} = D^{n_d}$.
 - Initialize the parameterized action-value NN $\hat{q}_{n_d}(x, a; \theta_{n_d})$:
 - For episode $e = 1$ to $MaxEpisodes$.
 - Reset environment and obtain initial observation: $x = [x_{o1}, x_{o2}, x_{o3}]$.
 - Select action a using epsilon-greedy policy.
 - The parameterized action-value is calculated for all actions.
 - Execute the selected action a by running the Simulink model using the Step function.
 - Calculate Episode Reward $R = fn(\text{error}_{\text{volume}_{\text{cm}3}})$ by execution the volume error script.
 - Update NN using loss function
 - Update the epsilon value for the next episode ϵ_{i+1}
 - Reset the environment to its initial state for the next episode.
 - II) Store the trained model $\hat{q}_{n_d}(s, a; \theta_{n_d})$: for each n_d .

4. End of Loop Over All Cases.

- Repeat training for all active valve configurations n .
 - 5. Output:
 - Return trained models for each number of active valves.
-

4.5. Scale script using sampled interaction cases across pressure variations

In RL, the agent learns from interaction with an environment where context ($Q_{\text{initial}}, Q_{\text{final}}, \Delta P$) is included in the observations. Limited environmental coverage can bias the agent actions and policy, leading to suboptimal decision-making in unseen context highlighting the need for diverse observations for robust generalization.

Due to inherent imbalances in interaction frequency, there were approximately 200 times more samples for $n_d = 1$ than for $n_d = 8$. Initially, the full interaction set was used for training (Section 4.4). However, to enhance computational efficiency and evaluate adaptability, only 10 % of the cases will be provided for the model to train on. Selection criteria included covering diverse flow ranges, maintaining representative distribution across all 2702 cases (Fig. 9), and prioritizing flow conditions with high "badness number" The badness number (BN) in DFCU measures the deviation in flow between two states, indicating how much the combined flow of initial and final states differs from the ideal, whether exceeding or falling short of the largest individual flow rate (Laamanen et al., 2007). While this ensured broad coverage, it may exclude rare transitions, limiting generalization. This reflects a common challenge in hydraulics where data collection is costly.

In Section 2.1.2, Fig. 5 showed that the initial and final states have a logical symmetry. Therefore, interaction selection can be based on either initial or final flow rates; in this case, the initial flow rate is used. This sampling strategy addresses imbalances across groups by independently

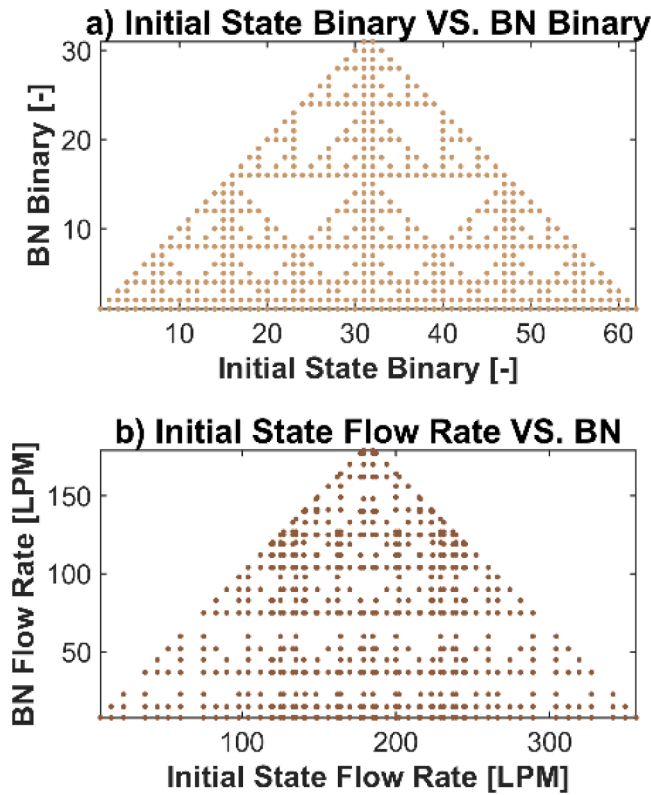


Fig. 9. Initial flow/states and badness number relation across 2702 cases for ideal and actual DFCU ($n = 6$). Final flow/states with BN exhibit exactly the same pattern.

sampling each group, preserving its structure. It prioritizes extreme boundary points and skews selection toward high badness scores and ensures a minimum sample size for smaller groups. This approach enhances scenarios diversity as seen in Fig. 10.

The following script trains the RL agent using a reduced subset of the data, focusing on interaction cases at ΔP 5 bar. The same subset size, adjusted for different flow rate outputs, is used for training at ΔP 10 and ΔP 20 bar.

DCB (ΔP -Loop) Algorithm: Training for Valve Delay Optimization with Selected Training Cases across different pressure values

1. Initialization:
 - Define 8 different active valve configurations ($n_d = 1$ to $n_d = 8$).
2. Main Loop: Over the active valves (for $n_d \leq 8$) :
 - a) Loop over: ΔP 5 bar, ΔP 10 bar, and ΔP 20 bar:
 - i) Prepare sampled interaction cases for training at each pressure level.
 - ii) Load the code in Section 4.4.
 - iii) Train the NN agent using the selected interaction scenarios.
3. End of Loop
 - After completing all active valve configurations and pressure levels.
4. Output:
 - Return 8 trained agents for each valve configuration n_d .

5. Comparisons and discussion

The trained agents are loaded alongside the Simulink model environment. Based on the current observation, which includes the initial flow rate, target final flow, and the pressure difference (ΔP), the appropriate agent generates artificial delay values for each active valve. Any external disturbance that modifies ΔP is captured in the observation before the delay decision is made, ensuring that the selected delays are optimal under the current load conditions. The system is sampled with a timing that reflects the physical response time of the valve; no new observation is taken until the valve has completed its transition. This constraint reflects real-world behavior and contributes to the robustness

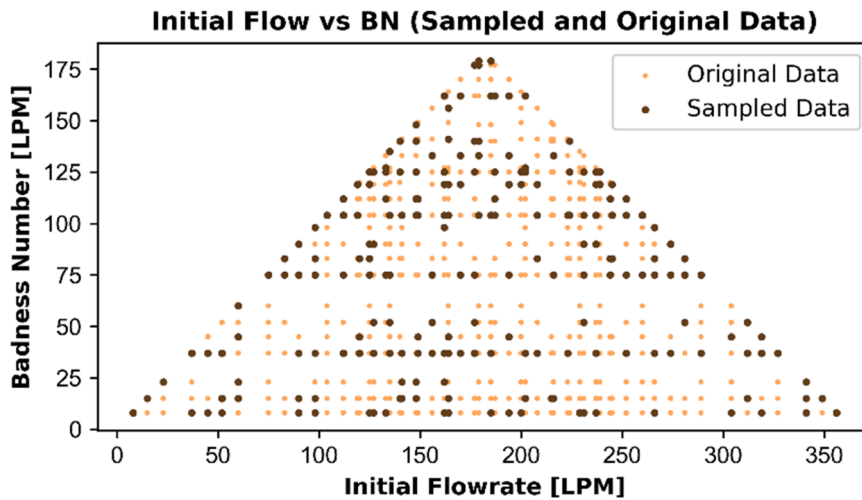


Fig. 10. One-tenth sampled scenarios captures extremes, balancing diverse flow and badness across 2702 Cases.

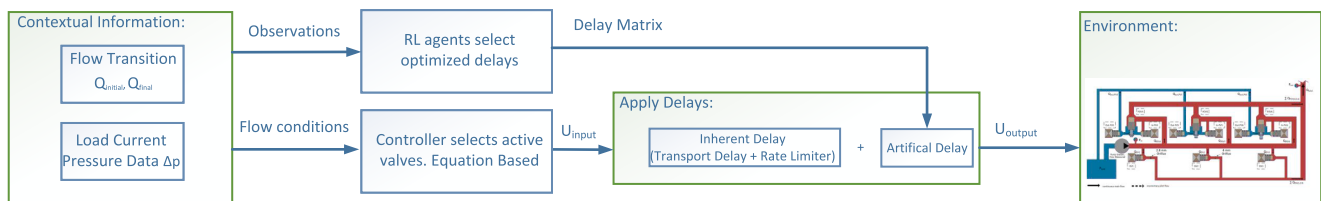


Fig. 11. Testing Framework for RL Valve Timing.

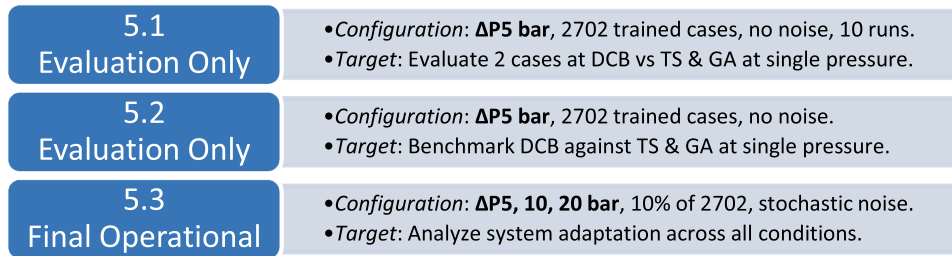


Fig. 12. Overview of experimental design and expected outcomes.

Table 3

Results of 10 repeated valve switching for different cases, Order: [DV1, DV2, DV3, In_PV4, Out_PV4, In_PV5, Out_PV5, In_PV6, Out_PV6].

Case	285			2672			DCB
	TS	GA	DCB	TS	GA	DCB	
Error [cm^3] No Optimization	0.60			4.47			
Avg. Error [cm^3]	0.32	0.32	0.32	0	0		0
Delays During Closing [ms]	[4]	[4]	[4]	[0,0,0,0,0,8,0,0,0]	[4,4,0,8,0,2,0,0,0] [4,0,4,8,2,4,2,0,0] [2,0,0,8,8,0,8,0,0] [8,0,0,2,0,0,8,0,0]	[0, 0, 0, 2, 0, 8, 2]	
Avg. Duration [sec]	0.7	0.8	9.3*	18	26		47*

* The reported training duration corresponds to the specific training case, which is a subset of the total training cases.

of the controller against sudden pressure variations. The deployment process for testing a single case is illustrated in Fig. 11.

All computations were performed on a machine with an Intel Core i7-1370P CPU at 2.40 GHz and 32 GB of RAM. Three distinct test scenarios were selected for algorithm analyzing as shown in Fig. 12:

5.1. Switching for minimal and maximum valve configurations at $\Delta P5$ bar

To ensure reliability, the models were executed 10 times under consistent settings for thorough evaluation. In the First scenario (Table 3), (Case 285 $\rightarrow n_d = 1$) DV3 is closed, and DV2 opens, changing the valve configuration from [0, 0, 1, 0, 0, 0] to [0, 1, 0, 0, 0, 0] and reducing flow rate from 37 LPM to 15 LPM. Classical optimization failed to meet the $\epsilon < 0.2cm^3$ inequality constraint, suggesting no feasible solution with the given actions. The DCB runtime increases with the 20-episode setting, as Simulink computational and memory demands grow with the complexity of agent-environment interactions.

Whereas (Case 2672 $\rightarrow n_d = 7$) this represents an extreme case where Valves DV1, DV2, DV3, POV4, and POV5 are shifted to closed, and Valve POV6 is switched open ([1, 1, 1, 1, 1, 0] to [0, 0, 0, 0, 0, 1]), reducing flow rate from 239 LPM to 125 LPM. Both classical algorithms achieved zero error, but GA randomness causes varying methods. The DCB possible action for this case, defined by $(OC \times D)^i$, yields 16,384 possible combinations, leading to longer runtimes due to the complex action set. The deterministic results in DCB arise because the ϵ -greedy policy transitions to near-pure exploitation as ϵ decays to its minimum value during training, combined with the deterministic nature of the environment and reward structure (noise will be implemented in Section 5.3).

5.2. Optimization over entire 2702 cases at single $\Delta P5$ bar

In the Second scenario, the best result from the 10 runs in Section 5.1 is plotted in Fig. 13, highlighting GA outperforming with lowest error and faster computation. TS achieves low errors but takes slightly longer, while DCB shows higher average errors. DCB average results could improve but risks outlier impact, as NN-based approach can yield

suboptimal solutions (Bengio et al., 2021).

TS and GA are employed **only** to assess the optimal solution targeted by the DCB model

The preliminary overall performance of the DCB model, as shown in Fig. 13, is then analyzed in greater detail in Fig. 14 by dividing it into 8 groups. The analysis highlights that $n_d 2$ and $n_d 4$ exhibit the highest error deviation, making them the most challenging to predict accurately, while $n_d 1$, $n_d 7$, and $n_d 8$ have the lowest deviation. Outliers are also observed, potentially impacting model stability and accuracy.

5.3. Extended pressure with representative scenarios set at $\Delta p 5, 10, 20$ bar with stochastic noise

Performance evaluation

Third Scenario: In this section, the algorithm outlined in Section 4.5 is applied to evaluate training performance. For each pressure level, a representative subset comprising one-tenth of the 2702 cases is used to simulate training under limited interaction conditions. The objective is to adapt the system to handle continuous pressure inputs across all cases. The addition of stochastic noise introduces challenges but is expected to enhance exploration during training.

Fig. 15 demonstrates the DCB model acceptable performance, though it faces challenges at unseen pressures (3 and 15 bar). Additionally, the presence of outliers highlight the need for additional constraints post-interaction to ensure robustness under new conditions (Bengio et al., 2021). While tailored stochastic NN adjustments improve training time to **7.8 hours** (Detailed in Table 4) across the Δp range, the process remains lengthy due to the complexity of stochastic NN in managing high-dimensional and dynamic environments (Fourati et al., 2024). Moreover, despite robust training outcomes, DCB underperforms in certain testing by selecting suboptimal delay values. This occurs due to “forgetting” as network overwrites previously learned action-values while adapting to new interactions, losing previously learned

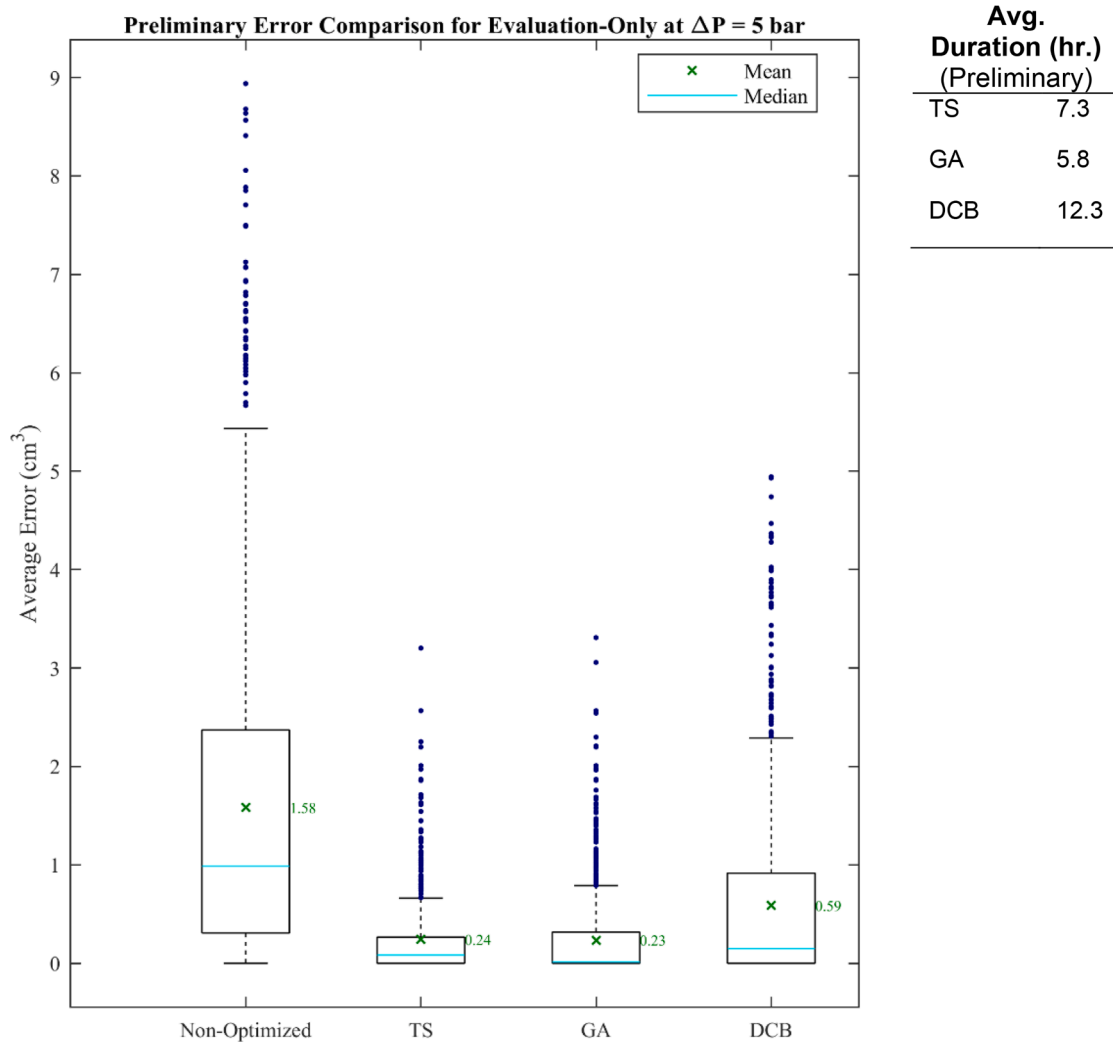


Fig. 13. Preliminary error comparison of Tabu search, genetic algorithm, and deep contextual bandit for DFCU. Trained and tested across 2702 cases for best model at ΔP 5 bar.

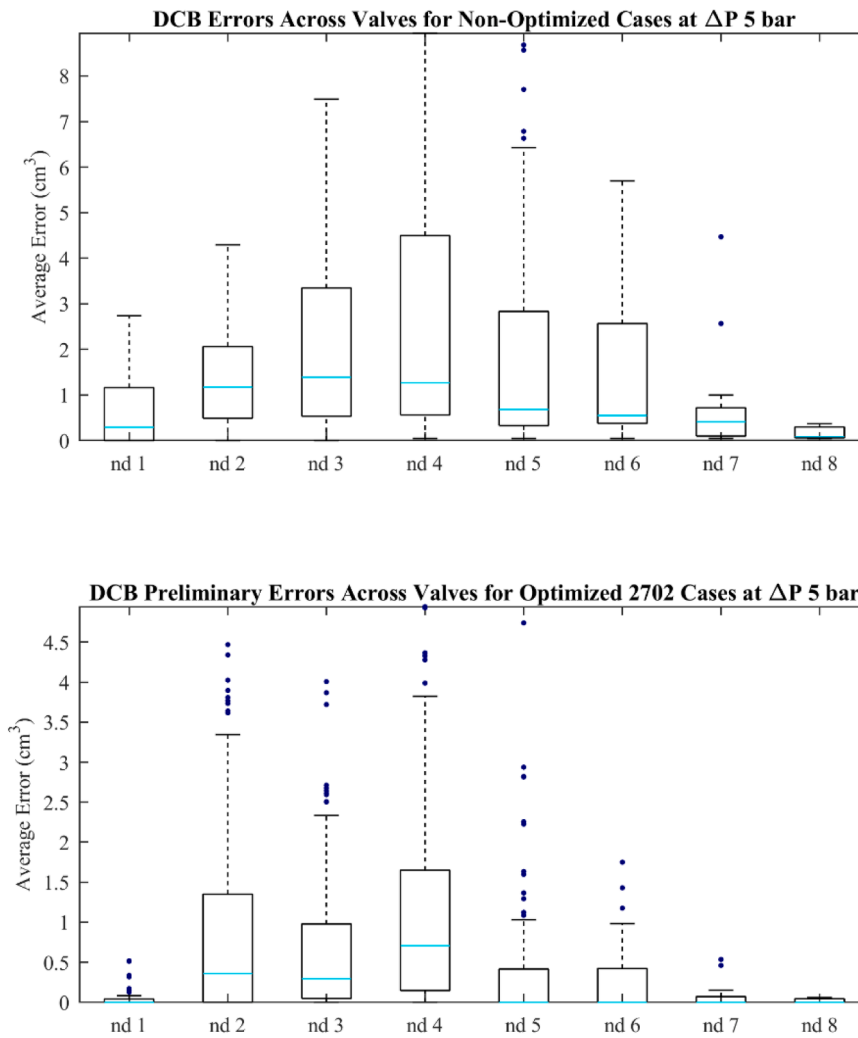


Fig. 14. DFCU Error comparison across valves for non-optimized and preliminary optimized 2702 cases at $\Delta P 5$ bar using deep contextual bandit.

information (Talaei Khoei et al., 2023), despite efforts to mitigate it using experience replay.

Computational demands of models

Assessing RL computational complexity is essential for future hardware deployment, with particular emphasis on training efficiency, inference process and memory footprint (Ivanov et al., 2025). Although deep models achieve competent accuracy, they frequently exceed these metrics, posing a serious challenge for industrial valve systems where fast response is critical. Table 4 below reports these values.

Table 4 shows that model size grows exponentially with valve count, yet inference latency stays below 1 ms up to $n_d = 6$. While training time is governed by the number of sampled scenarios and model size. Survey techniques (Roth et al., 2024) for model compression and faster inference approaches that, while promising, lie beyond the scope of this paper.

Prospective considerations for real-world implementation

Non-ideal flow grid already used: All simulations were carried out with realistic steady-state capacities and valve dynamic based on nine electrically operated Bucher valves tested by (Ketonen & Linjama, 2017), and catalogue of three pilot-operated Rexroth valves, resulting in a non-ideal flow grid matching the planned valve manifold. Moreover, a corresponding high-flow test bench has also been studied and validated in simulation (Elsaed & Linjama, 2024b).

Uncertainty limits: To minimize the integrated flow error, three

challenging hardware constraints apply:

- Fluid condition: Hold oil at a stable temperature to keep flow drift negligible.
- Flow measurement: Use a sensor precise enough that its error doesn't dominate the integrated flow error. A side note, pressure sensing may be preferable if flow meters respond too slowly.
- Valve-timing uncertainty: Direct-acting valves are consistent; pilot-operated valves need much tighter stroke-to-stroke control to stay within the error band.

6. Conclusion

This study is the first to apply RL techniques to address long-standing issues of flow and pressure fluctuations in Digital Hydraulics, by employing a Deep Contextual Bandit approach for a six-valve DFCU. The method reduced the median integrated flow errors by approximately 60 %, with training completed in about 8 hours despite limited interaction scenarios. However, challenges with outliers persist. Moreover, certain flow transition cases exhibited residual errors, underscoring the difficulty of completely eliminating errors given the current action set. Alternative methods like Tabu Search and Genetic Algorithms also failed to resolve these cases, suggesting that expanding the action space could help but at the cost of longer training times.

Managing multiple case-specific models is computationally demanding. Future efforts could focus on unifying these into a single

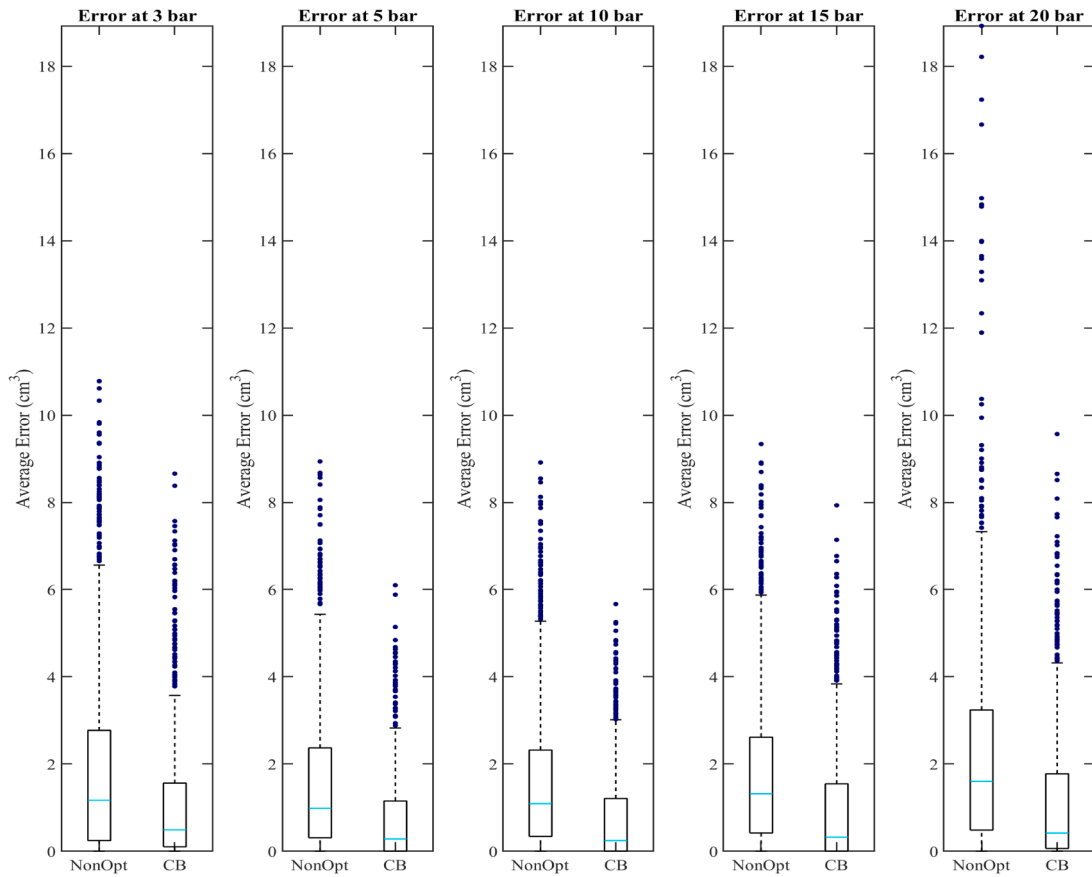


Fig. 15. Error distribution for DFCU Optimization at ΔP5, 10, 15, and 20 bar: DCB (10% Cases at 5, 10 and 20 bar; unseen at 3 and 15 bar).

Table 4
Computational cost as a function of valve count (n_d) for Section 5.3 trained agents.

n_d	Parameters No. (k)*	Memory Usage (MB)**	Sampled Scenarios	Total Episodes***	Training Time (h)	Inference Latency (ms)
	$P = \sum_{i=1}^L (I_i O_i + O_i)$	$M = \frac{P \times 4}{1024}$	S : Section 5 req.	$E_t = S \times E \times N_p$	$T_{train} = \Delta t_{train}$	$getAction (trainedAgent, obs)$
1	42	0.16	60	3600	0.8	0.7
2	43	0.17	80	4800	1.3	0.7
3	46	0.18	59	3540	0.6	0.7
4	59	0.23	35	2100	0.4	0.8
5	109	0.43	20	1200	0.3	0.8
6	309	1.21	20	1200	1.3	0.9
7	1107	4.32	17	1020	2.6	1.3
8	4302	16.80	3	180	0.5	1.4

PC Specs: Core i7–1370P CPU at 2.40 GHz and 32 GB of RAM.
 * L is the number of layers in the NN, where each layer I has $I_i O_i$ weights and O_i biases.
 ** MATLAB uses float32 by default: 4 bytes per parameter.
 *** $N_p = 3$ pressure levels and $E = 20$ Episodes/Scenario.

model to simplify training and decision-making. Exploring standard bandit algorithms, other single-step RL methods, and more sophisticated exploration strategies may enhance efficiency and adaptability. This study also revealed that system performance is sensitive to hyperparameter settings.

Overall, this study represents a significant foundational step toward achieving smoother transition states in digital hydraulic systems. It contributes to the growing integration of intelligent methods in hydraulic system design and control.

Acronyms

Acronym	Description
CB	Contextual Bandit.
DCB	Deep Contextual Bandit.
DFCU	Digital Flow Control Unit.
DV	Direct Valve.
GA	Generic Algorithm.
MHs	Metaheuristics.
MLP	Multilayer Perceptron.
PCM	Pulse Code Modulation.
POV	Pilot Operated Valve.

(continued on next page)

(continued)

Acronym	Description
PV	Pilot Valves.
RL	Reinforcement Learning.
TS	Tabu Search.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported partially by the Ministry of Education in Finland, represented in Tampere university Automation Technology and Mechanical Engineering unit (ATME), specifically within the Innovative Hydraulics and Automation (IHA) Lab. And partially by the Egyptian Cultural Affairs and Missions.

Data availability

Data will be made available on request.

References

- Ban, Y., Yuchen, Y., Banerjee, A., & He, J. (2022). 38979666 · EE-Net: exploitation-exploration neural networks in contextual bandits. *International Conference on Learning Representations*.
- Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2), 405–421.
- Chapple, P. (2014). *Principles of hydraulic systems design*. Momentum Press.
- Chen, J., Gao, Y., Kasihmuddin, M. S. M., Zheng, C., Romli, N. A., Mansor, M. A., Zamri, N. E., & When, C. (2024). MTS-PRO2SAT: hybrid mutation Tabu search algorithm in optimizing probabilistic 2 satisfiability in discrete hopfield neural network. *Mathematics*, 12(5), 721.
- Chenggang, Y., & Pan, M. (2024). AN integrated compact digital hydraulic converter prototype for hydraulic transmission. Tampere, Finland: The Twelfth Workshop on Digital Fluid Powe. <https://events.tuni.fi/uploads/sites/1247/1247/12/48b11a50-dfp24-proceedings.pdf>.
- Coskun, M. Y., & Itik, M. (2024). Position control of a digital electrohydraulic system with limited sensory data using double deep q-network controller. *Expert Systems with Applications*, 252, Article 124275.
- D'Angelo, G., & Palmieri, F. (2021). GGA: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems. *Information Sciences*, 547, 136–162.
- Elsaed, E., & Linjama, M. (2024a). Analyzing the design and performance of a high flow rate digital flow control unit: Structural insights and simulation findings. *Advances in Mechanical Engineering*, 16(9), Article 16878132241277338. <https://doi.org/10.1177/16878132241277338>
- Elsaed, E., & Linjama, M. (2024b). Development and simulation of a test system for analyzing high flow rate digital flow control units. In *12th Workshop on Digital Fluid Power 2024* (pp. 7–24). <https://researchportal.tuni.fi/en/publications/development-and-simulation-of-a-test-system-for-analyzing-high-fl>.
- Elsaed, E., & Linjama, M. (2024c). Dynamic response analysis of the main plunger in a two-stage on/off poppet valve for the digital hydraulics field. *International Journal of Fluid Power*, 325–348.
- Elsaed, E., & Linjama, M. (2024d). Evaluating the performance of machine learning CFD-based and hybrid analytical models for transient flow prediction in temperature-compensated digital flow units. *Flow Measurement and Instrumentation*, 95, Article 102511. <https://doi.org/10.1016/j.flowmeasinst.2023.102511>
- Fourati, F., Aggarwal, V., & Alouini, M.-S. (2024). Stochastic Q-learning for large discrete action spaces. In *Proceedings of the 41st International Conference on Machine Learning* (pp. 13734–13759). <https://proceedings.mlr.press/v235/fourati24a.html>.
- Fricke, C., Wolff, D., Kemmerling, M., & Elgeti, S. (2023). Investigation of reinforcement learning for shape optimization of profile extrusion dies. *Advances in Computational Science and Engineering*, 1(1), 1–35. <https://doi.org/10.3934/acse.2023001>
- Garmendia, A. I., Ceberio, J., & Mendiburu, A. (2022). *Neural combinatorial optimization: A new player in the field* (No. arXiv:2205.01356). arXiv <http://arxiv.org/abs/2205.01356>.
- Hafeez, M. A., Rashid, M., Tariq, H., Abideen, Z. U., Alotaibi, S. S., & Sinky, M. H. (2021). Performance improvement of decision tree: A robust classifier using tabu search algorithm. *Applied Sciences*, 11(15), 6728.
- Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). <https://ojs.aaai.org/index.php/AAAI/article/view/11694>.
- Hu, K., Li, M., Song, Z., Xu, K., Xia, Q., Sun, N., Zhou, P., & Xia, M. (2024). A review of research on reinforcement learning algorithms for multi-agents. *Neurocomputing*, Article 128068.
- Huova, M., & Linjama, M. (2022). Energy efficient throttling control of a multi-pressure system using a genetic algorithm and model predictive control. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 236(2), 406–417. <https://doi.org/10.1177/09596518211027713>
- Huova, M., Tammisto, J., & Linjama, M. (2020). Open-loop independent metering control of a multi-dof forwarder boom. *International Journal of Fluid Power*, 21(2), 147–168.
- Ivanov, D. A., Larionov, D. A., Maslennikov, O. V., & Voevodin, V. V. (2025). Neural network compression for reinforcement learning tasks. *Scientific Reports*, 15(1), 9718.
- Karamichailidou, D., Kaloutsas, V., & Alexandridis, A. (2021). Wind turbine power curve modeling using radial basis function neural networks and Tabu search. *Renewable Energy*, 163, 2137–2152.
- Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296(2), 393–422.
- Ketonen, M., & Linjama, M. (2017). High flowrate digital hydraulic valve system. *The ninth workshop on digital fluid power* (pp. 7–8).
- Kharrat, S., Fourati, F., & Canini, M. (2024). ACING: actor-critic for instruction learning in black-box large language models (No. arXiv:2411.12736). arXiv. <https://doi.org/10.48550/arXiv.2411.12736>.
- Kumagai, W., & Yasuda, K. (2023). Black-box optimization and its applications. In T. Kaihara, H. Kita, S. Takahashi, & M. Funabashi (Eds.), *Innovative systems approach for facilitating smarter world* (pp. 81–100). Nature Singapore: Springer. https://doi.org/10.1007/978-981-19-7776-3_6.
- Laamanen, A., Linjama, M., & Vilenius, M. (2005). *Pressure peak phenomenon in digital hydraulic systems-a theoretical study*. 91–104.
- Laamanen, A., Linjama, M., & Vilenius, M. (2007). On the pressure peak minimization in digital hydraulics. In *Tenth Scandinavian Conference on Fluid Power, SICFP'07*.
- Larsson, L. V., Lejonberg, R., & Ericson, L. (2022). Optimisation of a pump-controlled hydraulic system using digital displacement pumps. *International Journal of Fluid Power*, 53–78.
- Li, Y., Mu, Z., & Qi, S. (2024). A contextual combinatorial bandit approach to negotiation. In *Proceedings of the 41st International Conference on Machine Learning* (pp. 28448–28465). <https://proceedings.mlr.press/v235/li24az.html>.
- Li, Z., & Ai, Q. (2023). Managing considerable distributed resources for demand response: A resource selection strategy based on contextual bandit. *Electronics*, 12(13), 2783.
- Lu, W. (2020). *Review of the digital hydraulics technologies* [PhD Thesis, Politecnico di Torino] <https://webthesis.biblio.polito.it/16887/>.
- Luo, Y. (2006). *System modeling and control design of a two-stage metering poppet-valve system* [University of Missouri–Columbia] https://mospace.umsystem.edu/xmlui/handle/10355/4453?utm_source=chatgpt.com.
- Luo, Y., Wang, Y., Dong, K., Liu, Y., Sun, Z., Zhang, Q., & Song, B. (2023). SIRL: Self-imitation reinforcement learning for single-stage hitting tasks. In *2023 International Conference on Advanced Robotics and Mechatronics (ICARM)* (pp. 185–190). <https://ieeexplore.ieee.org/abstract/document/10218831/>.
- Mantovani, I. J., Kagueiama, H. A., Gama, A. T., de, C., Dell'Amico, A., Krus, P., & De Negri, V. J. (2020). On/off valves synchronization and reliability evaluation of a digital hydraulic actuator. *Fluid Power Systems Technology*, 83754. V001T01A041 <https://asmedigitalcollection.asme.org/FPST/proceedings-abstract/FPMC2020/83754/1088967>.
- Mazyavkina, N., Sviridov, S., Ivanov, S., & Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134, Article 105400.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., & Ostrovski, G. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Ozalp, A. F., Polat, R., Cetinkaya, C., & Cetin, M. H. (2021). Investigation of a digital hydraulic valve operated by servo motors. *Engineering, Technology & Applied Science Research*, 11(6), 7957–7963.
- Padovani, D., & Barth, E. J. (2018). Exploiting valve timing for pneumatic energy savings. *Fluid Power Systems Technology*, 51968. V001T01A037 <https://asmedigitalcollection.asme.org/FPST/proceedings-abstract/FPMC2018/V001T01A037/271113>.
- Pellikka, M., Ahola, V., Soederlund, L., & Uusitalo, J.-P. (2011). Genetic optimization of a fast solenoid actuator for a digital hydraulic valve. *International Journal of Fluid Power*, 12(2), 49–56. <https://doi.org/10.1080/14399776.2011.10781030>
- Raduenz, H., Ericson, L., De Negri, V. J., & Krus, P. (2022). Multi-chamber actuator mode selection through reinforcement learning—Simulations and experiments. *Energies*, 15(14), 5117.
- Raghuwanshi, P., L'opez, O., Mehta, N., & Latva-aho, M. (2024). Neural network-based bandit: A medium access control for the IIoT alarm scenario | IEEE Journals & Magazine | IEEE Xplore. <https://ieeexplore.ieee.org/abstract/document/10767384>.
- Renault, M., Viquerat, J., Meliga, P., Grandin, G.-A., Meynet, N., & Hachem, E. (2023). Investigating gas furnace control practices with reinforcement learning. *International Journal of Heat and Mass Transfer*, 209, Article 124147.
- Robbins, H. (1952). Some aspects of the sequential design of experiments. <https://projecteuclid.org/journals/bulletin-of-the-american-mathematical-society/volume-58/i>

- ssue-5/Some-aspects-of-the-sequential-design-of-experiments/bams/1183517370.pdf.
- Roth, W., Schindler, G., Klein, B., Peharz, R., Tschatschek, S., Fröning, H., Pernkopf, F., & Ghahramani, Z. (2024). Resource-efficient neural networks for embedded systems. *Journal of Machine Learning Research*, 25(50), 1–51.
- Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., & Wen, Z. (2018). A tutorial on Thompson sampling. https://web.stanford.edu/~bvr/pubs/TS_Tutorial.pdf.
- Sciatti, F., Tamburrano, P., Distaso, E., & Amirante, R. (2024). Digital hydraulic valves: Advancements in research. *Heliyon*. [https://www.cell.com/heliyon/fulltext/S2405-8440\(24\)03295-X](https://www.cell.com/heliyon/fulltext/S2405-8440(24)03295-X).
- Slivkins, A. (2019). *Introduction to multi-armed bandits*. Foundations and Trends® in Machine Learning. <https://doi.org/10.1561/22000000068>
- Stylianopoulos, K., Alexandropoulos, G., Huang, C., Yuen, C., Bennis, M., & Debbah, M. (2022). Deep contextual bandits for orchestrating multi-user MISO systems with multiple RISs. In *ICC 2022-IEEE International Conference on Communications* (pp. 1556–1561). <https://ieeexplore.ieee.org/abstract/document/9838369/>.
- Talaei Khoei, T., Ould Slimane, H., & Kaabouch, N. (2023). Deep learning: Systematic review, models, challenges, and research directions. *Neural Computing and Applications*, 35(31), 23103–23124. <https://doi.org/10.1007/s00521-023-08957-4>
- Valmet. (2016). Digital hydraulics, valmet technical paper series. https://www.valmet.com/globalassets/media/downloads/white-papers/process-improvements-and-parts/wpp_digihydraulics.pdf.
- Viquerat, J., Duvigneau, R., Meliga, P., Kuhnle, A., & Hachem, E. (2023). Policy-based optimization: Single-step policy gradient method seen as an evolution strategy. *Neural Computing and Applications*, 35(1), 449–467. <https://doi.org/10.1007/s00521-022-07779-0>
- Wang, C., Cheng, P., Li, J., & Sun, W. (2024). Leader reward for POMO-based neural combinatorial optimization (No. arXiv:2405.13947). arXiv <http://arxiv.org/abs/2405.13947>.
- Yang, J., Li, J., Zhong, Y., Gao, Y., Guo, R., & Zhao, J. (2023). Research on design and control strategy of novel independent metering system. *Sustainability*, 15(18), Article 13359.
- Zhou, D., Li, L., & Gu, Q. (2020). Neural contextual bandits with UCB-based exploration. In *Proceedings of the 37th International Conference on Machine Learning* (pp. 11492–11502). <https://proceedings.mlr.press/v119/zhou20a.html>.