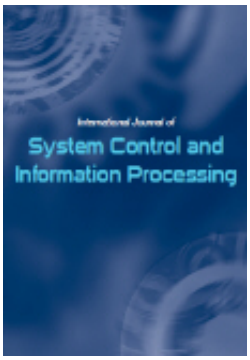


International Journal of System Control and Information Processing

ISSN online: 1759-9342 - ISSN print: 1759-9334
https://www.inderscience.com/ijscip

Dynamic resource adjustment of hierarchical control Petri nets

Yuxiao Zhu, Jinliang Xue

DOI: 10.1504/IJSCIP.2025.10069722

Article History:

Received:	26 December 2024
Last revised:	05 January 2025
Accepted:	06 January 2025
Published online:	27 June 2025

Dynamic resource adjustment of hierarchical control Petri nets

Yuxiao Zhu*

Department of Automation,
Shanghai Jiao Tong University,
Shanghai, 200240, China
Email: zyx0321@sjtu.edu.cn
*Corresponding author

Jinliang Xue

National Security Internet Co., Ltd.,
Ningbo Artificial Intelligence Institute,
Shanghai Jiao Tong University,
Ningbo, 315000, China
Email: xuejinliang@sjtu-naii.com

Abstract: Efficient resource allocation is critical in water treatment systems to ensure reliable operation and manage anomalies. This study proposes a flow allocation algorithm based on hierarchical control Petri nets. Through dynamic rate adjustment and integration of boundary constraints and priority strategies, it achieves resource optimization. A hierarchical control mechanism is also established for resource scheduling in hybrid systems.Using a dynamic water supply system as an example, experiments validate the algorithm's applicability in scenarios like liquid level overflow and insufficient flow. Results show that the system restores liquid levels effectively while ensuring high-priority resource needs, providing a comprehensive solution for resource management and anomaly handling in water treatment systems.

Keywords: Petri nets; hybrid systems; hierarchical control; priority Allocation; dynamic flow rate allocation.

Reference to this paper should be made as follows: Zhu, Y. and Xue, J. (2025) 'Dynamic resource adjustment of hierarchical control Petri nets', *Int. J. System Control and Information Processing*, Vol. 4, No. 3, pp.253–272.

Biographical notes: Yuxiao Zhu obtained her Bachelor's degree in Automation from Shanghai Jiao Tong University and is currently pursuing a Master's degree in the Department of Automation at the same university. Her research interests include formal analysis and modelling, as well as industrial control system security.

Jinliang Xue is an Engineer at the Ningbo Artificial Intelligence Institute of Shanghai Jiao Tong University. His research interests include industrial control system security and industrial control networks.

1 Introduction

In modern industrial control systems (ICS), water treatment plants serve as a critical link in the water supply chain, with their operational stability and efficiency directly affecting the normal functioning of the economy and society. However, due to factors such as external environmental changes and improper resource allocation, water treatment systems frequently face anomalies such as level overruns and insufficient flow.

Challoumis (2024) stated that insufficient flow is typically caused by external environmental changes or internal system failures. Traditional anomaly analysis methods mainly include rule-based monitoring systems and preset threshold alarm strategies. However, Oberascher et al. (2022) pointed out that water treatment systems often involve nonlinear relationships and complex interactions. Rule-based systems, which rely on simple or single combination rules to detect anomalies, tend to have high false alarm and missed detection rates when dealing with complex data. Umer et al. (2020) proposed a method that uses association rule learning to identify invariants in control layer designs. However, this method relies on expert knowledge to convert continuous input data into discrete states.

In recent years, Chandola et al. (2009) attempted to apply machine learning methods to anomaly detection and analysis in water treatment systems to improve detection accuracy and responsiveness. Supervised learning methods train classifiers using historical data to identify anomaly patterns. Das et al. (2020) proposed an anomaly detection method that trains system norm classification models through logical data analysis. However, anomaly data in water treatment systems are often scarce or lack precise labels. To address issues of data labelling and generalisation, unsupervised learning methods have been increasingly applied in anomaly analysis. Methods such as clustering (Pang et al., 2021) and statistical modelling (Inoue et al., 2017) have been proposed to distinguish normal from abnormal states, mitigating data labelling problems to some extent. However, Sommer and Paxson (2010) noted that unsupervised methods suffer from poor interpretability and limited generalisation capabilities, making them ineffective at identifying new anomalies.

In summary, while machine learning methods have significantly improved anomaly detection accuracy, their "black-box" nature and dependence on high-quality data limit their widespread application in water treatment systems. As a result, Adepu and Mathur (2018) and García Soto et al. (2019) advocated for a shift towards learning system normative models instead of traditional black-box models. Researchers have begun exploring formal models based on Petri nets. As a powerful tool, Petri nets can describe and study concurrency, asynchrony, distribution, parallelism and randomness in information processing (Murata, 1989).

However, due to the interaction between discrete events and continuous dynamics in water treatment systems, traditional Petri net modelling methods often struggle to handle complex concurrent behaviours and dynamic changes effectively. To address this, hybrid Petri net models were proposed and developed by David and Alla (1994). Hybrid Petri nets (HPN) have been shown by Ghasemieh et al. (2016) and Jongerden et al. (2016) to be effective in assessing the reliability of critical infrastructure. Gribaudo and Remke (2016)

resolved conflicts in fluid flow through the use of sharing and prioritisation. Cao et al. (2018) introduced a hybrid stochastic timed Petri net (HSTPN) model to describe hybrid systems with discrete, continuous, conflicting, delayed, and stochastic characteristics. Hussain et al. (2023) proposed a data-interpreted Petri net (DIPN) hybrid modelling approach for anomaly detection in ICS. Cao et al. (2022) introduced a method based on an improved Mixed hybrid stochastic timed Petri net (M-HSTPN) to address complex situation representation and strategy reasoning problems in hybrid game systems. Li et al. (2023) proposed an integrated method for scheduling and control of manufacturing systems based on Parallel Petri Nets, aiming to eliminate the gap between the scheduling layer and the control layer.

Building upon this work, this paper proposes a dynamic rate adjustment algorithm based on hierarchical control Petri net modelling. The system is divided into control and simulation layers, which coordinate through upper and lower-level interactions. The algorithm dynamically adjusts flow rates based on system priority rules to address anomalies. This approach effectively smooths variable transitions, enhancing system stability and responsiveness.

The organisation of this paper is as follows: Sections 2 and 3 introduce the structure of hierarchical control algorithms and the operational mechanisms of Hierarchical Control Petri Nets. Sections 4 and 5 propose the dynamic resource adjustment algorithm. Section 6 validates the effectiveness of the proposed algorithm in scenarios of liquid level control and priority allocation through experiments. Finally, Section 7 concludes the study.

2 Construction of hierarchical control algorithms for Petri nets

2.1 Definition of hybrid Petri nets

A HPN is a type of Petri net model that combines discrete events and continuous events. It is suitable for describing complex systems that involve both discrete state changes and continuous dynamic processes. A HPN is formally defined as a seven-tuple:

$$Q = \{P, T, A, W, m_0, h, K(p)\} \tag{1}$$

- $P = \{P_1, P_2, \ldots, P_n\}$ is a finite and non-empty set of places, where $P = P^C \cup P^D$, P^C represents continuous places, and P^D represents discrete places.
- $T = \{T_1, T_2, \ldots, T_m\}$ is a finite and non-empty set of transitions, where $T = T^C \cup T^D$, T^C represents continuous transitions, and T^D represents discrete transitions.
- $P \cap T = \emptyset$, meaning P and T are disjoint sets.
- $A \subseteq (P \times T) \cup (T \times P)$, the set of arcs, representing the connections between places and transitions.
- $W : A \to \mathbb{N}$ assigns a weight to each arc, representing the relationship or influence between places and transitions.
- $m_0 : P \to \mathbb{R}^+$ or $\mathbb{N}$, assigns an initial value to each place.
- $h : P \cup T \to \{D, C\}$, known as the "hybrid function", which indicates whether a node is a discrete node (comprising P^D and T^D) or a continuous node (comprising P^C and T^C).

- $K(p) : P \to \mathbb{R}^+$ or $\mathbb{N}$, sets an upper limit for each place.

A HPN is an extended modelling framework that combines discrete and continuous variables, making it suitable for representing dynamic behaviours in complex systems. It is particularly effective for modelling critical infrastructures, such as water treatment plants, where both discrete state changes (e.g., switching equipment on or off) and continuous dynamics (e.g., flow rates) must be captured.

2.2 Components of hierarchical control Petri nets

A guard condition $G(t)$ is defined as a Boolean logical expression associated with a transition $t \in T$, satisfying the rule that the firing condition for transition t is:

$$\forall p \in \cdot t,\ M(p) \geq W(p,t) \text{ and } G(t) = \texttt{true} \tag{2}$$

- $\cdot t$ is the set of input places for transition t
- $M(p)$ is the current marking of place p
- $W(p,t)$ is the weight of the arc from place p to transition t
- $G(t)$ is the guard condition. Transition t can only fire if $G(t)$ evaluates to `true`.

The mathematical form of the guard condition $G(t)$ is expressed as:

$$G(t) : \Phi(M,E) \to \{\texttt{true}, \texttt{false}\}, \tag{3}$$

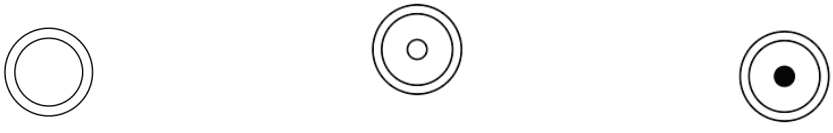
where $\Phi(M,E)$ is a Boolean expression of the current marking M and events E.

In hierarchical control Petri nets, continuous places can evolve through three states based on the differential equation process, as shown in Figure 1. These states include inactive, active but not enabled, and enabled states.

- Inactive state refers to when the physical process has not started, and the continuous process is in a dormant state.
- Active but not enabled state refers to when the physical process has started, the continuous variable is dynamically evolving, and it is approaching the triggering condition.
- Enabled state refers to when the state of the continuous place meets the guard condition, triggering the control place and enabling the related transition.

A decision place is a special type of place in a hierarchical control Petri net that enforces system behaviour constraints and manages states.The representation of a decision place in a Petri net is shown in Figure 2. A decision place can only be activated when specific conditions are met. Its state directly influences the firing of related transitions. The design of decision places allows for dynamic adjustments to adapt to the changing demands of complex hybrid systems.

Figure 1 Three states of continuous places: (a) inactive; (b) active but not enabled and (c) enabled



(a) (b) (c)

Figure 2 Two states of decision place: (a) decision place and (b) place activation

(a) (b)

The activation rules of the decision place are as follows.

$$m(p^{\text{dec}}) = \begin{cases} 1, & \text{if the activation condition is satisfied} \\ 0, & \text{if the activation condition is not satisfied} \end{cases} \tag{4}$$

- When $m(P_c) = 1$, it indicates that firing is allowed.
- When $m(P_c) = 0$, it indicates that firing is prohibited.

In the Petri net model, synchronised transition pairs are used to ensure synchronised operations between the upper control layer and the lower simulation layer. The synchronised transition pair is defined as follows:

$$T_{\text{sync}} = \{t_{\text{up}}, t_{\text{low}}\} \tag{5}$$

where,

- t_{up} is the transition in the upper control layer, responsible for monitoring system states and determining transition firing.
- t_{low} is the transition in the lower simulation layer, which is triggered simultaneously when the upper layer transition fires, executing state transitions.

2.3 Definition of hierarchical control Petri net

The hierarchical control Petri net divides the hybrid system into a control layer and a simulation layer. The control layer is modelled as a Petri net structure, responsible for determining the system's operating state and making decisions. The lower simulation layer uses differential equations to simulate the actual operation of the water treatment system, receiving decisions from the upper logical control layer, updating states, and synchronising the system's evolution with the upper layer.

A hierarchical control Petri net can be defined as a seven-tuple:

$$HCPN = (P, T, A, M_0, G, J, F) \tag{6}$$

where:

- $P = P^C \cup P^D \cup P^{\text{dec}}$: A set of places, including continuous places P^C, discrete places P^D, and decision places P^{dec}.
- $T = T^C \cup T^D$: A set of transitions, including continuous transitions and discrete transitions.
- $A \subseteq (P \times T) \cup (T \times P)$: A set of arcs, representing the connections between places and transitions.
- M_0: The initial marking, defining the token distribution in places at the initial state of the system.
- $G : T \to \{\text{true}, \text{false}\}$: A guard function, specifying the firing conditions for each transition.
- J: A mapping function that associates each lower-layer transition with an expression. When the corresponding transition is fired, the system's evolution rules transform according to the connected expression.
- F: A set of differential equations defining the dynamic changes of continuous state variables $p \in P^C$, including conditions such as thresholds and transition states under which these equations apply.

The upper control layer determines whether to fire a transition t_{up} based on the lower-layer state and Petri net logic, generating a control signal $C(t)$. The lower simulation layer simulates real-time states through differential equations, synchronises states with the upper control layer, receives the control signal $C(t)$, fires the transition t_{low} and updates the state.

The state update equations are expressed as follows:

$$M'(p) = \begin{cases} M(p) - W(p, t_{\text{up}}), & \text{if } p \in {\cdot}t_{\text{up}} \\ M(p) + W(t_{\text{up}}, p), & \text{if } p \in t_{\text{up}}{\cdot} \\ M(p), & \text{otherwise} \end{cases} \tag{7}$$

$$M'(t) = M(t_0) + \int_{t_0}^{t} (q_{\text{in}}(\tau) - q_{\text{out}}(\tau))\,\mathrm{d}\tau, \quad \text{when the relevant conditions are satisfied.} \tag{8}$$

- $M(p)$ and $M(t_0)$ are the markings of discrete and continuous places, respectively. For discrete places, the marking follows the standard Petri net transition firing logic. For continuous places, the marking evolves according to the transition firing logic defined by J through corresponding differential equations.
- $M'(p)$ and $M'(t)$ are the new markings after transition firing and state evolution.

In the hierarchical control Petri net, the upper logical controller determines the firing of transitions based on the lower-layer states and transfers them to the lower simulation layer for state evolution via synchronised transition pairs. Through the above formalised definitions, the operational mechanism of the hierarchical control Petri net can be clearly described, ensuring real-time synchronisation and control between the upper logical controller and the lower simulation layer.

3 Operation algorithms of hierarchical control Petri net

Algorithms 1 and 2 demonstrate the execution process of the hierarchical control Petri net, with a focus on the coordinated interaction between the control layer and the simulation layer. In the control layer, the status of places is monitored to determine whether to activate the decision place, triggering the corresponding synchronised transition (e.g., starting or stopping a pump), updating system markings, and sending control signals $C(t)$ to the lower-level executor. Upon receiving the control signal, the executor evaluates the actual simulation state to determine whether the synchronisation conditions are met and then triggers the corresponding synchronised transition. During execution, the lower layer continuously updates the state and sends the system state $F(t)$ back to the upper layer, ensuring real-time monitoring and adjustment of the control logic.

Algorithm 1: Upper Logical Control Layer Algorithm

Input: Initial markings M_0, guard conditions $G = \{G(t_{\text{start}}), G(t_{\text{stop}})\}$, system state $F(t)$ from the lower simulation layer
Output: Updated markings M_{up}

```
Initialize M_up ← M_0;
while system is active do
    Evaluate guard conditions G = {G(t_start), G(t_stop)};
    if P ∈ ·t is enabled and P^dec is active then
        Trigger the synchronized transition t_up;
        Update control layer markings M_up ← f(M_up, F(t));
        Send control signal C(t);
    else
        if P^C is active but not enabled then
            Send control signal C(t);
            Continue simulation layer evolution
        end
    end
    F(t) ← Receive state from the lower simulation layer;
end
```

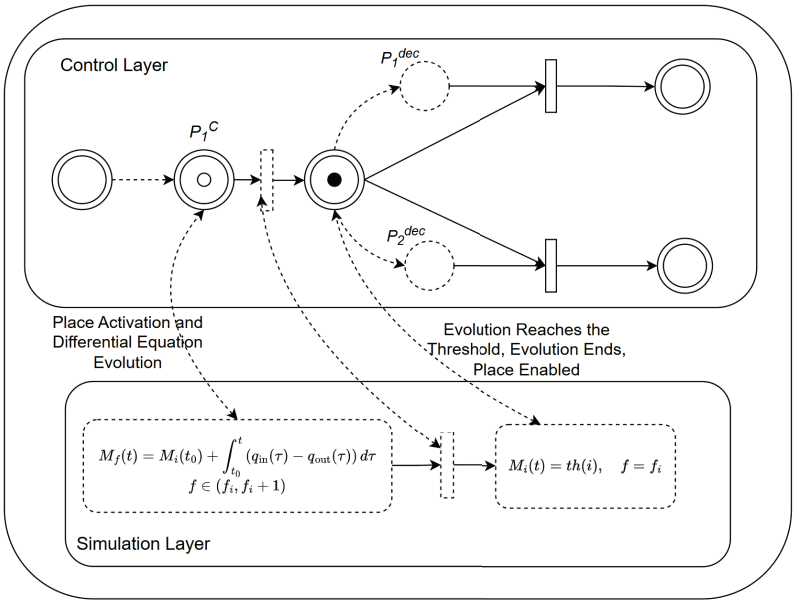
The task of the upper layer is global scheduling. By monitoring feedback signals $F(t)$, it evaluates whether the simulation layer satisfies the guard conditions. When the conditions are met and the decision place is activated, it triggers the synchronised transition t_{sync} and sends a start command $C(t)$ to the lower-level executor. The simulation layer focuses on simulating the system's real-time state. Upon receiving the upper layer's control signal to trigger lower-layer transitions, it updates its execution logic and reports the task completion status to the upper layer.

Algorithm 2: Lower-Level Simulation Layer Algorithm

Input: Control signal $C(t)$
Output: Updated state $F(t)$
while *system is active* **do**
  if $C(t) =$ *triggers synchronized transition* t_{low} **then**
    Trigger the synchronized transition t_{low};
    Update $\{F\}$ expressions;
    Update simulation layer state;
  end
end
Synchronize system state $F(t)$ with the upper logical control layer;

The entire process achieves seamless integration between the upper and lower layers through synchronised transitions and decision places. An example is shown in Figure 3. The continuous place P_1^C is described by a differential equation during its activation and gradual evolution. When the liquid level in the place reaches the upper or lower boundary, different decision places are activated, triggering different transitions to protect the system.

Figure 3 The hierarchical control Petri net architecture diagram



4 Definition of dynamic resource adjustment algorithm and safety measures

This section primarily discusses the measures to address risks when the system detects fluid levels exceeding upper or lower limits. The focus is on using a rate adjustment algorithm to regulate places reaching their boundaries, along with guard mechanisms and test arc mechanisms to ensure system safety.

4.1 Definition of liquid level abnormalities

The actual flow rate v_i represents the rate of change of fluid tokens m_i in the fluid place P_i over time t. It is typically calculated based on the inflow and outflow rates of the place, as follows:

$$v_i = \frac{dm_i}{dt} = \sum_{T_j \in \cdot p} V(T_j) - \sum_{T_k \in p \cdot} V(T_k) \tag{9}$$

where $V(T_j)$ is the actual flow rate of the fluid transition T_j.

For each continuous place $P_j^C \in P_C$ with a non-zero flow rate ($v_j \neq 0$), the function $\tau(P_j^C)$ specifies the remaining time before the place $P_j^C \in P_C$ reaches its boundary. It is calculated as follows:

$$\tau(P_j^C) = \begin{cases} \left| \frac{m_j}{v_j} \right| & \text{if } v_j < 0, \\ \frac{K(P_j^C) - m_j}{v_j} & \text{if } v_j > 0. \end{cases} \tag{10}$$

When $\tau_P(P_j^C) < \delta$, the continuous place is considered to be at risk of a liquid level abnormality. At this point, the rate adjustment algorithm is triggered to ensure system safety.

4.2 Definition of dynamic rate adjustment algorithm

When a continuous place is at risk of a liquid level abnormality, the system's flow rates need to be redistributed. The core idea of flow redistribution is to handle places near their boundaries by sequentially allocating flow rates based on priority. High-priority transitions are prioritised, and the remaining flow rates are distributed according to the arc weights and proportional shares.

Priority represents the level of precedence of an arc between a place and a transition during resource allocation.

$$R(P_i^C, T_j^C) = r_{i,j} \tag{11}$$

- $R(P_i^C, T_j^C)$ denotes the priority of the arc between the continuous place P_i^C and the transition T_j^C.
- $r_{i,j}$ is a non-negative integer representing the priority value of the arc (P_i^C, T_j^C). A smaller value of $r_{i,j}$ indicates a higher priority.

The rate adjustment algorithm iteratively allocates new actual flow rates to all enabled fluid transitions connected to a place based on their priorities.

After enabling flow rate adjustment, the algorithm first considers all enabled transitions connected to the place through arcs with priority l. It accumulates their nominal flow rates and calculates the flow demand $\omega_l(p_i^c)$ for the place p_i^c under priority l:

$$\omega_l(p_i^c) = \sum_{T_j^C \in T^C \cap T_j^C \text{ enabled in state } \Gamma} \eta(T_j^C) \cdot \rho_l(P_i^C, T_j^C) \tag{12}$$

where:

- $\eta(T_j^C)$ is the nominal flow rate, representing the maximum flow rate the transition can transfer under ideal conditions.
- $\rho_l(P_i^C, T_j^C)$ defines the connection priority and weight allocation between the place and the transition, as defined below:

$$\rho_l(P_i^C, T_j^C) = \begin{cases} W(P_i^C, T_j^C) & \text{if } \tau < \delta \cap v_i < 0 \cap (P_i^c, T_j^C) \in A^C \cap R(P_i^C, T_j^C) = l \\ W(T_j^C, P_i^C) & \text{if } \tau < \delta \cap v_i > 0 \cap (T_j^C, P_i^C) \in A^C \cap R(T_j^C, P_i^C) = l \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

The following discussion only considers the lower boundary case. The handling of the upper boundary is identical, with only the arc direction reversed. Let $\alpha(P_i^C)$ represent the available flow rate of place P_i^C, i.e., the actual flow rate that can currently be allocated by the continuous place. Starting from the highest priority $l = l_{\max}$, the available flow is allocated, with $\alpha_{\ell\max}(P_i^C) = \alpha(P_i^C)$. If $\alpha_l(P_i^C) \geq \omega_l(P_i^C)$, all transitions with priority l can be triggered at their nominal rates. The remaining flow rate for priority $l-1$ is calculated as follows:

$$\alpha_{l-1}(P_i^C) = \alpha_l(P_i^C) - \omega_l(P_i^C) \tag{14}$$

When the priority level $n \leq l$ is reached and the available flow is insufficient to meet the demand, the flow rate is allocated according to priority and weight:

$$V(T_j^C) = \beta_n(T_j^C) \cdot \frac{\alpha_n(P_i^C)}{W(P_i^C, T_j^C)} \tag{15}$$

- $\beta_n(T_j^C)$: The allocation ratio for transition T_j^C at priority n.
- $W(P_i^C, T_j^C)$: The weight of the connecting arc.

For demands with priority $R < n$, the flow rate is set to 0. The flow allocation process is then complete.

After completing the flow allocation, the actual flow rate $V(T_j^C)$ for the continuous transition T_j^C is calculated as:

$$V(T_j^C) = \min\left(\eta(T_j^C), \alpha(P_i^C)\right) \tag{16}$$

- $\eta(T_j^C)$: The nominal flow rate of transition T_j^C. This represents the maximum achievable flow rate under ideal operating conditions without any constraints.
- $\alpha(P_i^C)$: The current available rate of place P_i^C. This value is dynamically computed based on the liquid level state at the place, reflecting the maximum flow rate allowed by the guard conditions.

This formula ensures that the actual flow rate $V(T_j^C)$ of a transition cannot exceed its nominal flow rate or the maximum flow rate allowed by the current state of P_i^C. The guard conditions directly influence the calculation of $\alpha(P_i^C)$, dynamically constraining the flow rate allocation process.

4.3 System boundary processing

When allocating rates, it is essential to fully consider the system's boundary conditions to ensure its safety. The handling of boundary conditions mainly relies on the combined use of test arcs, inhibitor arcs, and guard conditions. Test arcs can monitor the current capacity of fluid places to determine whether transitions meet triggering conditions; inhibitor arcs prevent illegal flows when places reach boundaries (such as full or empty) through restrictions on specific fluid conditions; guard conditions set logical constraints for transitions, ensuring flow is only triggered when specific conditions are met.

These mechanisms work together to dynamically adjust flow rates in fluid allocation, preventing place overflow or depletion, thus achieving safe and stable system operation. Here are the definitions first.

In HPN, a test arc is a special arc that checks whether the marking of the input place meets specific conditions without changing the place's marking. Let Γ be a state of the HPN. Let (P_i, T_j) represent a test arc from place P_i to transition T_j, the test arc enabling condition can be defined as:

$$T_j \text{ enabled } \Longrightarrow m(P_i) \geq W(P_i, T_j) \tag{17}$$

- $m(P_i)$ is the marking of place P_i;
- $W(P_i, T_j)$ is the weight of the test arc.

The function of a test arc is to ensure that transition T can only be enabled when the number of tokens $m(P)$ in the place is greater than or equal to the required minimum number of tokens $W(P_i, T_j)$. The main characteristic of test arcs is that they do not change the number of tokens in places, but are only used to detect whether conditions are met, thus constraining transition triggering. An example of a test arc is shown in Figure 4(a), where the valve closing transition can only be triggered when the resources in P_2 reach 15, thus ensuring the safety of the fluid level in place P_2.

Inhibitor arcs are used to inhibit transition triggering. An inhibitor arc allows its connected transition to be activated if and only if the number of tokens in the input place is less than a given threshold.

Assume system state Γ. For an inhibitor arc $\langle P_i, T_j \rangle_h$ from place P_i to transition T_j, it is defined as:

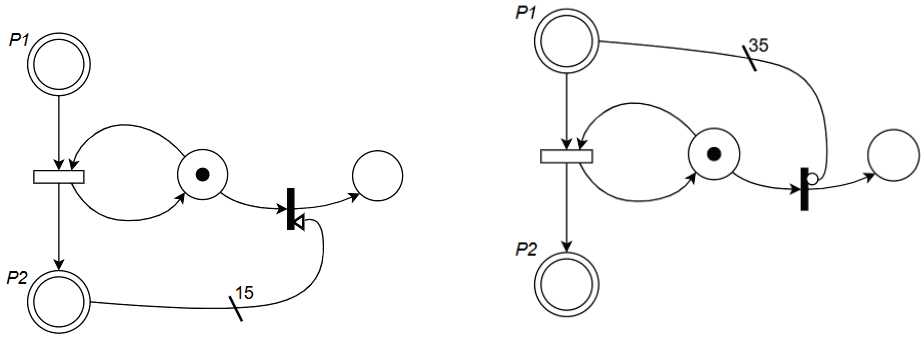
$$T_j \text{ enabled } \Longrightarrow m(P_i) < W(P_i, T_j) \tag{18}$$

where:

- $m(P_i)$ is the marking of place P_i;
- $W(P_i, T_j)$ is the weight of the inhibitor arc.

This formula indicates that transition T_j is forbidden to fire when the number of tokens in P_i is greater than or equal to the inhibitor arc weight $W(P_i, T_j)$. An example of an inhibitor arc is shown in Figure 4(b), where the valve is prohibited from closing $m(P_1) > 35$.

Figure 4 Examples of test and inhibitor arcs: (a) test arc and (b) inhibitor arc



(a) (b)

In water treatment systems, test arcs and inhibitor arcs are typically used together to achieve more fine-grained control. For example: test arcs are used to detect if the liquid level has reached the lower limit, if so, triggering the water replenishment transition; inhibitor arcs are used to restrict water replenishment operations when the liquid level exceeds the upper limit, ensuring system safety. Meanwhile, during rate adjustment, inhibitor arcs provide forced protection to prevent liquid levels from exceeding boundaries, with the main protective measures as follows:

- *Upper bound protection*:
  When $x_i \geq K(P_i)$, inhibitor arcs disable all incoming transitions:
  $V(T_j^C) = 0$ (disable inflow operations).
- *Lower bound protection*:
  When $x_i \leq 0$, inhibitor arcs disable all outgoing transitions:
  $V(T_j^C) = 0$ (disable outflow operations).

The definition of guard conditions has already been provided earlier. Together, guard conditions and inhibitor arcs use logical constraints and firing restrictions to effectively ensure the safe operation of the system. Guard conditions ensure that transitions are fired only when the system is in a safe state. Inhibitor arcs prevent transitions from firing under specific conditions, thus avoiding risks such as overflow or depletion by ensuring fluid levels do not exceed or fall below critical thresholds.

4.4 Iterative adjustment algorithm for flow rate allocation

Each redistribution of fluid changes the inflow and outflow rates across the entire system, potentially affecting the flow rates of other fluid places. Therefore, when a liquid level abnormality occurs, an iterative adjustment algorithm is used to determine a stable solution for flow rates and to redistribute fluid. This algorithm performs iterative checks on all continuous places at their boundaries. The algorithm is shown in Algorithm 3.

Algorithm 3: Iterative Adjustment Algorithm for Flow Rate Allocation

Input: Fluid places P^C, fluid transitions T^C, initial flow rates v_0, guard conditions G
Output: Stable flow rate distribution θ

Initialize the drift v_i of all places $P_i \in P^C$;
Initialize the actual flow rates $V(T_j), \forall T_j \in T^C$;
repeat
  for $\forall p_i$ *where* $v_i > 0 \cap \tau < \delta$, **do**
    Call the rate adjustment algorithm;
    Update the drift v_i of place P_i;
  end
  for $\forall p_i$ *where* $v_i < 0 \cap \tau < \delta$, **do**
    Call the rate adjustment algorithm;
    Update the drift v_i of place P_i;
  end
  foreach *place* $p_i \in P^C$ **do**
    Verify guard conditions G;
    Compute the drift of the current place p_i: $v_i = \sum_{T_j \in \cdot p} V(T_j) - \sum_{T_k \in p \cdot} V(T_k)$
  end
until *The drift of all places* P_i *satisfies the boundary conditions*;
return *Stable flow rate distribution* θ

In each iteration, for all places at their upper or lower boundaries, the dynamic adjustment algorithm is applied to adjust the flow rates and update the drift of the places. The iteration terminates when the allocated flow rates of all fluid places at the upper boundary are less than or equal to zero, and the allocated flow rates of all fluid places at the lower boundary are greater than or equal to zero.

5 Dynamic adjustment algorithm for hierarchical control Petri nets

The boundary conditions and rate adjustment algorithm have been introduced earlier. When liquid levels reach their boundaries, the specific hierarchical control abnormality handling algorithm is described below.

5.1 Upper-level controller abnormality handling algorithm

When an abnormality is triggered, the upper-level controller adjusts the system using the rate adjustment algorithm and boundary conditions. In the upper-level controller, a Petri net model is constructed with the introduction of a decision place. In a water plant system, upon detecting that a liquid level has reached the threshold, the adjustment place is activated. After the decision place is activated, it enables transitions based on the rate adjustment algorithm and disables transitions triggered by nominal flow rates. This allows the system to redistribute flow rates according to the rate adjustment mode. Using a fixed-point algorithm, when the liquid levels of places return to normal, the decision place is disabled. Once the rate adjustment is completed, the system returns to its normal state. The modelling algorithm for the upper-level control layer is shown in Algorithm 4.

Algorithm 4: Upper-Level Controller Abnormality Handling Algorithm

Input: Fluid places P^C, fluid transitions T^C, boundary conditions $K(P^C)$, initial flow rates V_0
Output: Optimized flow rate distribution θ
Initialize liquid level states m_i for all places $P_i \in P^C$;
Initialize actual flow rates $V(T_j), \forall T_j \in T^C$;
repeat
  foreach *place* $P_i \in P^C$ **do**
    Monitor liquid level m_i;
    if $m_i \geq K(P_i)$ **then**
      Disable all inflow transitions;
    else
      if $m_i \leq 0$ **then**
        Disable all outflow transitions;
      else
        Adjust flow rates for all places at boundaries;
        foreach *transition* $T_j \in T^C$ **do**
          Call the iterative adjustment algorithm to redistribute flow rates based on priority and weight:

$$V(T_j^C) = \beta_n(T_j^C) \cdot \frac{\alpha_n(P_i^C)}{W(P_i^C, T_j^C)}$$

        end
        Dispatch the optimized flow rates θ and system states to the lower-level simulation layer;
      end
    end
  end
until *The liquid levels of all places* P_i *satisfy the boundary conditions*;
return *Optimized flow rate distribution* V

In the adjustment process of the upper-level HPN, E is used to represent the symbol for triggering events, and the types of triggering events are as follows:

- Discrete markings change, triggering elements $E_{i+1} \in T_D$.
- Continuous markings reach the boundary value, triggering elements $E_{i+1} \in P_C$.
- Changes in the continuous place markings cause guard conditions to be satisfied or not satisfied, triggering elements $E_{i+1} \in A_T$.
- When a continuous place reaches its boundary, the rate adjustment algorithm is used to redistribute flow, triggering elements $E_{i+1} \in P_C$.

5.2 Lower-level actuator abnormality handling algorithm

The lower-level controller encompasses the state of each place at different time steps during its evolution. In a water plant system, the following states are included:

1 $M(P_i^C)$: The liquid level state of place i.

2 The liquid level variation at each place P_i^C, where the change in liquid level m_i is determined by the drift v_i:

$$\frac{dm_i}{dt} = v_i = \sum_{T_j^C \in \cdot P_i^C} V(T_j^C) \cdot W(T_j^C, P_i^C) - \sum_{T_j^C \in P_i^C \cdot} V(T_j^C) \cdot W(P_i^C, T_j^C)$$

where:

- $\cdot P_i^C$: The set of inflow transitions connected to place P_i^C.
- $P_i^C \cdot$: The set of outflow transitions connected to place P_i^C.
- W: The weight of the connecting arcs.
- $V(T_j^C)$: The adjusted flow rate determined by the control layer.

3 $th(P_i^C)$: The threshold of place i.

The differential equations at the lower level are responsible for simulating the system's state. When the system reaches a critical state, it triggers transitions in the upper layer. Once the upper-layer transitions are triggered, the differential equations corresponding to the relevant places start evolving.

6 Experimental validation

Based on the aforementioned theories and methods, this section will experimentally verify the feasibility and effectiveness of the hierarchical control abnormality handling algorithm.

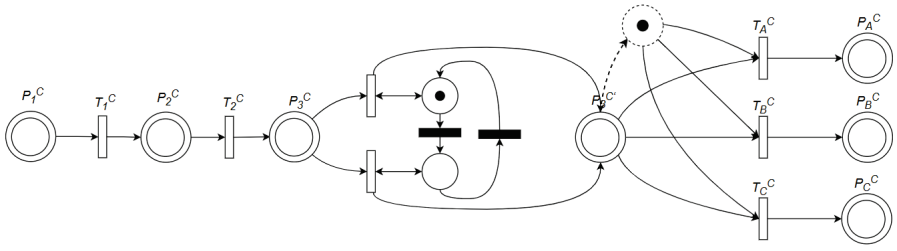
6.1 Introduction to the dynamic water supply system

As shown in Figure 5, the system represents a water plant. The system consists of three core stages: the softening stage (P_1^C), the filtration stage (P_2^C), and the storage stage (P_3^C). The softening stage has a large storage capacity but a slower processing rate. Water flows through transition T_1^C to the filtration stage, with flow rates limited to prevent overflow in the filtration stage. The filtration stage has a smaller storage capacity but a faster processing rate. Water flows through transition T_2^C to the storage stage, ensuring stable water supply. The storage stage is used for the final storage of processed water, which is distributed to consumers of different priorities via transitions T_A^C, T_B^C, and T_C^C.

Day-night demand is controlled by two transitions: during the daytime (6:00–21:00), water is replenished rapidly at a rate of 2 to quickly refill the reservoir; during the nighttime (21:00–6:00), water is replenished at a slower rate of 1. In the consumer prioritisation strategy, Consumer A (e.g., hospitals) receives water via T_A^C from P_3^C and has the highest priority; Consumer B (e.g., industrial facilities) receives water via T_B^C and has the second-highest priority; and Consumer C (e.g., residential areas) receives water via T_C^C with the lowest priority. When the storage level is insufficient, priority is given to ensuring the water supply for Consumer A, while the supply for lower-priority consumers may be reduced or

halted. The system dynamically adjusts the flow rates of the softening and filtration stages to ensure efficient operation during day-night transitions and to meet the demands of critical users.

Figure 5 Water supply system Petri net



The hierarchical control schematic of the water plant system is shown in Figure 6. During the simulation, the main triggering events include

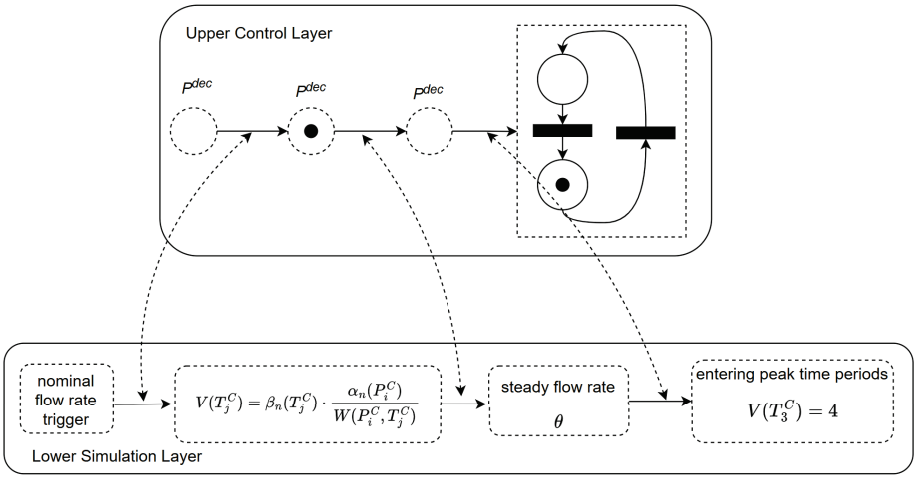
1 rate adjustment

2 reaching safe flow rates

3 entering peak time periods.

Triggering events are primarily reflected in the upper layer as token transitions in the Petri net and in the lower layer as changes in simulation strategies.

Figure 6 Hierarchical control of water treatment plant systems

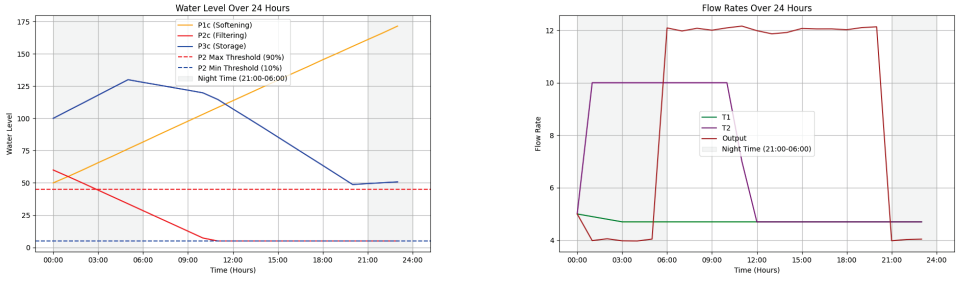


6.2 Validation of liquid level exceeding boundaries

The experiment simulated scenarios where the liquid level of P_1^C was too low and the liquid level of P_2^C exceeded its upper limit. From the liquid level variation diagrams shown in

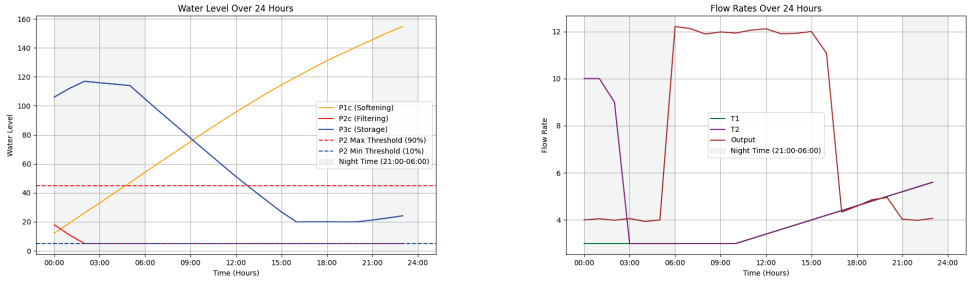
Figures 7 and 8, it can be observed that the Petri net-based hierarchical control system effectively captures and adjusts abnormal fluctuations in liquid levels. The experimental results demonstrate that when the liquid levels exceed the upper or lower boundaries of the places, the control algorithm promptly adjusts the flow rates, gradually bringing the liquid levels back to the safe range. The liquid level of P_1^C gradually rises and stabilises within the safe range. Meanwhile, the liquid levels of P_2^C and P_3^C remain within reasonable ranges, showing the system's strong self-regulation capabilities. Particularly during nighttime periods (shaded area), when the output flow rate decreases, all three tanks exhibit relatively stable liquid levels.

Figure 7 Liquid level and rate changes when P_2^C exceeds upper limit: (a) liquid level and (b) rate adjustment (see online version for colours)



(a) (b)

Figure 8 Liquid level and rate changes: (a) liquid level and (b) rate adjustment (see online version for colours)



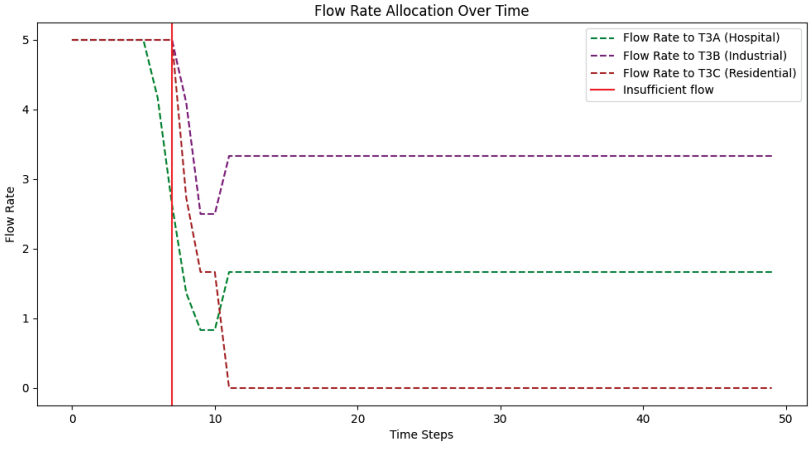
(a) (b)

The rate adjustment diagrams further validate the real-time adjustment capability of the Petri net hierarchical control system. Upon detecting deviations from the safe range, the system dynamically adjusts flow rates to ensure that liquid levels quickly return to the target range. The rate adjustments maintain smooth transitions, avoiding new abnormalities caused by over-adjustment, which reflects the flexibility and stability of the control logic. These experimental results fully demonstrate the applicability of this control model in water plant systems, providing robust theoretical and practical support for enhancing the safety and stability of ICSs.

6.3 Validation of priority-based allocation during insufficient flow

This experiment demonstrates the dynamic adjustment process of water flow allocation under different conditions in a multi-priority system. As shown in Figure 9, in the initial phase, the flow is allocated in a $2:6:4$ ratio, reflecting balanced supply for various water usage demands. At $T = 8$, by observing the system behaviour at key transition points, a flow shortage is detected, requiring the priority algorithm to reallocate the flow. When $T > 10$, the system quickly transitions to a $2:4:0$ allocation strategy, ceasing supply to the lowest-priority user (T_3^C). This adjustment ensures that resources are prioritised for higher-priority users in the event of potential shortages. The experimental results clearly visualise the system's responsiveness and resource redistribution efficiency under predefined conditions, validating its applicability in scenarios with resource fluctuations.

Figure 9 Priority-based allocation during insufficient flow (see online version for colours)



7 Conclusion

This paper proposes an anomaly handling algorithm for water plant systems based on a hierarchical control Petri net, aiming to address common issues such as liquid level overflow, insufficient flow rate, and priority allocation in water plant systems. Firstly, a hierarchical control Petri net model is established, and operating strategies are defined to achieve collaboration between continuous and discrete elements as well as between upper and lower layers. Additionally, a rate adjustment algorithm is introduced, combined with boundary condition constraints and priority allocation strategies, to dynamically optimise resource allocation. Moreover, considering the hierarchical structure of the system, this study implements a collaborative anomaly handling mechanism between the control layer and the simulation layer, effectively balancing the complexity and real-time performance of anomaly handling.

The experimental validation section uses a dynamic water supply system as an example to simulate operating scenarios under liquid level overflow and insufficient flow conditions. The experimental results demonstrate that the proposed anomaly handling

algorithm can effectively regulate flow rates and validate the effectiveness of the priority allocation strategy. In the future, this method could be extended to more complex industrial domains, such as intelligent manufacturing systems, energy distribution networks, and traffic management systems.

References

Adepu, S. and Mathur, A. (2018) 'Distributed attack detection in a water treatment plant: Method and case study', *IEEE Transactions on Dependable and Secure Computing*, Vol. 18, No. 1, pp.86–99.

Cao, R., Hao, L., Bai, G. and Gao, Q. (2018) 'Characteristic analysis of hybrid stochastic timed Petri net', *Proceedings of the 2018 Chinese Control and Decision Conference (CCDC)*, IEEE, Shenyang, China, pp.3955–3960.

Cao, R., Yang, H., Cui, J., Hao, L. and Wang, L. (2022) 'Situation representation and strategic reasoning method of hybrid game system based on modified hybrid stochastic timed Petri net', *IEEE Systems Journal*, Vol. 16, No. 4, pp.6086–6096.

Challoumis, C. (2024) 'Building a sustainable economy-how AI can optimize resource allocation', *Proceedings of the XVI International Scientific Conference*, Philadelphia, pp.190–224.

Chandola, V., Banerjee, A. and Kumar, V. (2009) 'Anomaly detection: A survey', *ACM Computing Surveys*, Vol. 41, No. 3, pp.1–58

Das, T.K., Adepu, S. and Zhou, J. (2020) 'Anomaly detection in industrial control systems using logical analysis of data', *Computers and Security*, Vol. 96, pp.101935

David, R. and Alla, H. (1994) 'Petri nets for modeling of dynamic systems: a survey', *Automatica*, Vol. 30, No. 2, pp.175–202.

García Soto, M., Henzinger, T.A., Schilling, C. and Zeleznik, L. (2019) 'Membership-based synthesis of linear hybrid automata', in Dillig, I. and Tasiran, S. (Eds.): *Computer Aided Verification, Lecture Notes in Computer Science*, Vol. 11561, Springer, New York City, NY, USA, pp.297–314.

Ghasemieh, H., Remke, A. and Haverkort, B.R. (2016) 'Survivability analysis of a sewage treatment facility using hybrid Petri nets', *Performance Evaluation*, Vol. 97, pp.36–56.

Gribaudo, M. and Remke, A. (2016) 'Hybrid Petri nets with general one-shot transitions', *Performance Evaluation*, Vol. 105, pp.22–50.

Hussain, M., Fidge, C., Foo, E. and Jadidi, Z. (2023) 'Discovering a data interpreted Petri net model of industrial control systems for anomaly detection', *Expert Systems with Applications*, Vol. 230, pp.120511.

Inoue, J., Yamagata, Y., Chen, Y., Poskitt, C.M. and Sun, J. (2017) 'Anomaly detection for a water treatment system using unsupervised machine learning', *Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, New Orleans, LA, USA, pp.1058–1065.

Jongerden, M.R., Huls, J., Remke, A. and Haverkort, B.R. (2016) 'Does your domestic photovoltaic energy system survive grid outages?', *Energies*, Vol. 9, No. 9, pp.736.

Li, D., Luo, J., Sun, S., Nie, W., Nie, Z. and Fang, H. (2023) 'Integrated scheduling and control method for manufacturing systems based on parallel Petri nets', *Acta Automatica Sinica*, Vol. 49, No. 4, pp.845–856.

Murata, T. (1989) 'Petri nets: Properties, analysis and applications', *Proceedings of the IEEE*, Vol. 77, No. 4, pp.541–580.

Oberascher, M., Rauch, W. and Sitzenfrei, R. (2022) 'Towards a smart water city: A comprehensive review of applications, data requirements, and communication technologies for integrated management', *Sustainable Cities and Society*, Vol. 76, pp.103442.

Pang, G., Shen, C., Cao, L. and Van Den Hengel, A. (2021) 'Deep learning for anomaly detection: a review', *ACM Computing Surveys*, Vol. 54, No. 2, pp.1–38

Sommer, R. and Paxson, V. (2010) 'Outside the closed world: on using machine learning for network intrusion detection', *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, Oakland, California, USA, pp.305–316.

Umer, M.A., Mathur, A., Junejo, K.N. and Adepu, S. (2020) 'Generating invariants using design and data-centric approaches for distributed attack detection', *International Journal of Critical Infrastructure Protection*, Vol. 28, pp.100341