Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

Learning-Based Peak Temperature Optimization for Multicore Pipelined Hard Real-Time Systems

Qiangxiao Zhou[1], **Hangbin Xu**[1], **Yiheng Wang**[1], **Long Cheng**[2,1]

[1] Wenzhou University
[2] Sun Yat-sen University
q.x.zhou.work@gmail.com, xu.hangbin@outlook.com, heng980526@163.com, chenglong3@mail.sysu.edu.cn

Abstract

Thermal-timing coupling in multicore pipelined systems makes it difficult to reduce peak temperature while satisfying hard real-time constraints. Periodic Thermal Management (PTM), which cyclically switches cores into low power mode, offers a deterministic and analyzable control strategy for thermal management in multicore processors. While existing PTM algorithms based on threshold or heuristic methods simplify the problem into a linearly constrained optimization for theoretical analysis, thereby neglecting the inherently nonlinear nature of the original optimization problem. To address this, we propose Autonomous Learning PTM (ALPTM), an online framework based on TD3 that formulates PTM optimization as a continuous-action Markov decision process, explicitly modeling the nonlinear coupled dynamics between thermal evolution, service curves, and queueing behavior. Experiments on representative streaming applications demonstrate that ALPTM consistently preserves hard real-time correctness and achieves significantly lower peak temperatures compared to existing PTM-based methods.

Introduction

In recent years, multicore processors have become the dominant architecture for hard real-time systems due to increasing computational demands. Pipeline execution, which allows multiple subtasks to run concurrently, significantly enhances throughput and is therefore widely adopted in embedded real-time platforms. However, ensuring hard real-time correctness under pipelined execution is increasingly challenging. Continuous growth in chip power density leads to elevated peak temperatures, threatening system reliability through timing drift, accelerated aging, and emergency throttling (Chantem, Dick, and Hu 2008). These effects are unacceptable in hard real-time systems; reducing peak processor temperatures while preserving real-time guarantees is a critical issue in modern multicore pipelines.

Processor temperature is closely determined by both dynamic and leakage power. Dynamic Thermal Management (DTM) techniques such as Dynamic Voltage and Frequency Scaling (DVFS) adjust supply voltage and frequency to control dynamic power (Dey et al. 2022; Yao 2023). Dynamic Power Management (DPM) reduces leakage power by transitioning cores into low power states (Benini, Bogliolo, and De Micheli 2000; Narang et al. 2023; Khan et al. 2023). As technology advances into nanometer regimes, leakage power becomes a dominant contributor to total power consumption. This trend makes DPM techniques more suitable for thermal regulation. Periodic Thermal Management (PTM) (Cheng et al. 2015; ul Islam et al. 2018; Cheng et al. 2021), a representative DPM technique, periodically alternates cores between $active$ and $sleep$ modes. Its deterministic structure makes it particularly attractive in hard real-time systems, yet designing effective PTM schemes that suppress peak temperatures while ensuring end-to-end real-time correctness remains challenging.

Integrating thermal management into pipelined scheduling introduces substantial complexity. A core's $active$ or $sleep$ duration directly shapes its service curve, which affects the queueing delay of upstream subtasks. These upstream delays propagate through the pipeline, influencing downstream workloads and modifying overall service demands. Meanwhile, temperature evolves slowly due to thermal inertia and exhibits spatial coupling across cores through heat diffusion (Huang et al. 2009). This coupling means that adjusting one core's temperature influences neighboring cores. As a result, PTM scheme decisions produce multi-cycle, cross-core interactions rather than independent local effects (Cheng et al. 2021).

These coupled dynamics imply that PTM is inherently a sequential decision-making problem defined in a continuous action space. Each PTM scheme decision influences future temperature trajectories, service capacities, and queue lengths. These factors collectively determine whether future events can meet their deadlines. This structure aligns naturally with the formulation of a Markov Decision Process (MDP), where system states evolve across cycles under the combined effects of thermal-timing dynamics (Gill et al. 2020; Zhou et al. 2021; Ilager, Ramamohanarao, and Buyya 2020). Traditional threshold-based methods, static schedules, and heuristic pattern selection struggle to capture these cross-cycle dependencies or adapt to fluctuating workloads. Consequently, there is a strong need for an online adaptive mechanism that can adjust PTM schemes in real-time while respecting hard constraints.

This paper proposes Autonomous Learning PTM

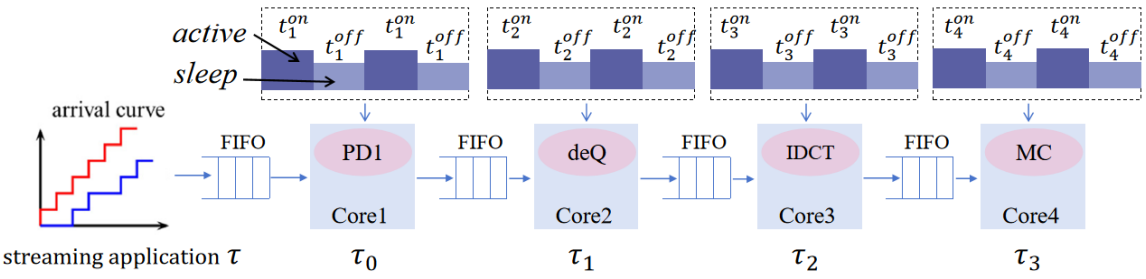


Figure 1: Periodic thermal management of H.263 decoders in pipeline hardware architectures.

(ALPTM), a TD3-based online framework that formulates PTM scheme optimization as a continuous-action MDP. By exploiting runtime temperature trends, service-curve variations, and latency backlogs, ALPTM learns adaptive *sleep*/*active* schedules capable of responding to dynamic execution conditions while preserving real-time feasibility. The key contributions are as follows:

- We formulate peak temperature minimization under hard real-time constraints as a continuous-action MDP, providing a more rigorous framework that captures the nonlinear coupled dynamics, thereby avoiding the limitations of linear approximation methods.
- We propose ALPTM, an online framework based on the TD3 algorithm. By leveraging the inherent nonlinear representational capacity of neural networks, ALPTM directly learns adaptive PTM schemes that capture the complex, coupled dynamics of thermal evolution, service curve variations, and queueing behavior in real-time.
- We evaluate ALPTM on multiple streaming applications, demonstrating that it maintains strict deadline compliance while achieving lower or comparable peak temperatures than representative PTM-based methods.

System Model

Hardwork Model

In pipeline-based multicore architectures, applications are decomposed into multiple subtasks and mapped to different cores. Communication between cores occurs sequentially via FIFO buffers, as shown in Figure 1. Each core operates in two processing modes: *active* or *sleep*, exhibiting significantly different power consumption levels. Moreover, switching between these modes incurs non-negligible time overhead (Mohaqeqi, Kargahi, and Fouladi 2016). Let t_i^{swon} and t_i^{swoff} denote the time overhead required for the i-th core to switch its mode to *active* and *sleep*, respectively. The magnitude of these values primarily depends on the hardware circuitry and driver software. During mode switching, power consumption equals that of the *active* mode, and no tasks can be processed. Due to the existence of mode switching time overhead, the duration that core i spends in *active* or *sleep* mode must be greater than the time required to switch to *active* or *sleep* mode, respectively, i.e. $t_i^{\text{on}} > t_i^{\text{swon}}$, $t_i^{\text{off}} > t_i^{\text{swoff}}$ (Mulas et al. 2009).

Application Model

This paper considers a streaming application τ traversing an n-core pipeline system. It imposes a hard real-time constraint, namely a relative end-to-end deadline D. We assume τ has been pre-partitioned into n sub-task streams, where τ_i denotes the sub-task executed on the i-th core, as shown in Figure 1. Its Worst-Case Execution Time (WCET) is denoted as c_i (Huang et al. 2012).

Assume the stream application generates an infinite sequence of events, where the release time of the j-th event τ^i is denoted as $r^j (j \in N^+)$. As previously described, each event τ^j should be partitioned into n sub-events τ_i^j, where i denotes the sub-event executed on the i-th core. In the pipeline architecture, the computational output of a sub-event serves as input to the subsequent sub-event. In other words, sub-event τ_i^j depends on the preceding sub-event $\tau_{i-1}^j (i \geq 2)$.

To satisfy the hard real-time constraint, the sub-event τ_n^j on the last core must complete before the absolute deadline $r^j + D$. That is:

$$F^j \leq r^j + D \tag{1}$$

where F^j is the completion time of sub-event τ_n^j. Thus, the hard real-time constraint for all events can be expressed as:

$$F^j - r^j \leq D, j \in N^+ \tag{2}$$

where $F^j - r^j$ is the delay of event τ_n^j, comprising two parts: execution time on the core and waiting time in the FIFO.

The arrival of stream application τ is described using the general model arrival curve $\alpha(\Delta) = [\alpha^u(\Delta), \alpha^l(\Delta)]$ (Hettiarachchi, Fisher et al. 2013). Here, $\alpha^u(\Delta)$ and $\alpha^l(\Delta)$ denote the upper and lower arrival curves, respectively, representing the upper and lower bounds of $R(t)$:

$$\alpha^u(\Delta) \geq R(t) - R(s) \geq \alpha^l(\Delta), \forall t - s = \Delta \tag{3}$$

where $R(t)$ is the cumulative load function denoting the number of events arriving within $[0, t]$. The arrival curve typically abstracts fundamental characteristics of many task timing models, such as periodic, sporadic, and those involving non-deterministic timing behavior.

Additionally, service curve $\bar{\beta}_i(\Delta)$ is used to model the available resources of core i during any time interval Δ (Yun, Shin, and Wang 2011). Similarly, $\bar{\beta}_i^u(\Delta)$ and $\bar{\beta}_i^l(\Delta)$

represent the upper and lower bounds of $C_i(t)$, where $C_i(t)$ denotes the number of time slots during which i-th core can service incoming events. Since the aforementioned arrival curve is event-based, the service curve is also converted to an event-based representation for convenience. The specific calculation is as follows:

$$\beta_i^u(\Delta) = \left\lfloor \frac{\bar{\beta}_i^u(\Delta)}{c_i} \right\rfloor, \beta_i^l(\Delta) = \left\lfloor \frac{\bar{\beta}_i^l(\Delta)}{c_i} \right\rfloor \tag{4}$$

For a single-core system to provide hard real-time guarantees, the following condition must hold:

$$\beta_i^l(\Delta) \geq \alpha^u(\Delta - D), \forall \Delta \geq 0 \tag{5}$$

For multi-core systems, the PBOO principle introduces the concept of an aggregated service curve (Cheng et al. 2016). For an n-core pipeline system to provide hard real-time guarantees, the following condition must hold:

$$\begin{aligned} \beta_{\text{tdma}}^l(\Delta) &= \beta_1^l(\Delta) \otimes \beta_2^l(\Delta) \otimes ... \otimes \beta_n^l(\Delta) \\ &\geq \alpha^u(\Delta - D), \forall \Delta \geq 0 \end{aligned} \tag{6}$$

Twin Delayed Deep Deterministic Policy Gradient

The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm (Fujimoto, Hoof, and Meger 2018) consists of Actor and Critic networks, each with an online and a target counterpart, resulting in six networks in total: the Actor network $\mu(\cdot|\theta^\mu)$, the Target Actor network $\mu'(\cdot|\theta^{\mu'})$, two Critic networks $Q_1(\cdot|\theta^{Q_1})$, $Q_2(\cdot|\theta^{Q_2})$, and their corresponding target networks $Q_1'(\cdot|\theta^{Q_1'})$, $Q_2'(\cdot|\theta^{Q_2'})$. The Actor is updated by maximizing the cumulative expected return, while the two Critics are updated by minimizing the temporal-difference error. All target networks are updated via soft updates.The training procedure is as follows:

First, interaction data in the form of $(s, a, r, s', \text{"done"})$ are collected and stored in a replay buffer. During learning, a mini-batch is sampled. The target action a' for the next state s' is computed using the Target Actor with added clipped noise:

$$a' = \mu'(s'|\theta^{\mu'}) + \epsilon, \quad \epsilon \sim \text{clip}(N(0, \sigma), -c, c) \tag{7}$$

Then, the target Q-value is obtained from the minimum of the two Target Critics. The Critic parameters are updated by minimizing the loss:

$$L_{c_i} = \left(Q_i(s, a|\theta^{Q_i}) - \left(r + \gamma \min_{i=1,2} Q_i'(s', a'|\theta^{Q_i'}) \right) \right)^2 \tag{8}$$

After every d updates of the Critics, the Actor is updated. The Actor computes a new action $a_{\text{new}} = \mu(s|\theta^\mu)$, which is evaluated by one of the Critics, e.g., $q_{\text{new}} = Q_1(s, a_{\text{new}}|\theta^{Q_1})$. The Actor parameters are then adjusted via gradient ascent to maximize q_{new}.

Finally, all target networks are softly updated by a rate psi:

$$\begin{aligned} \theta^{Q_i'} &= \psi\theta^{Q_i} + (1 - \psi)\theta^{Q_i'} \quad (i = 1, 2) \\ \theta^{\mu'} &= \psi\theta^{\mu} + (1 - \psi)\theta^{\mu'} \end{aligned} \tag{9}$$

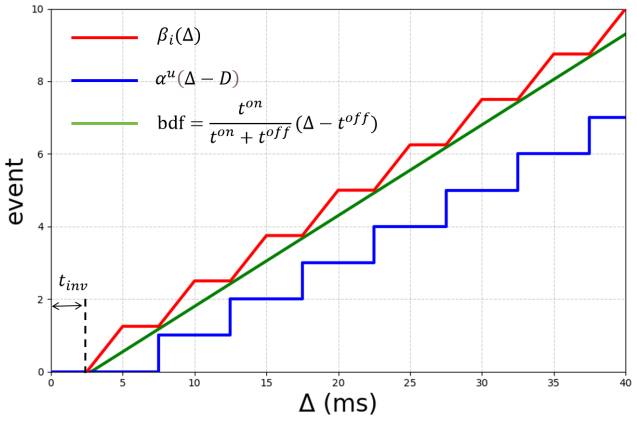


Figure 2: Service curve (red curve) of a PTM scheme lower-bounded by the BDF (greed curve), and the real-time constraint curve (in blue).

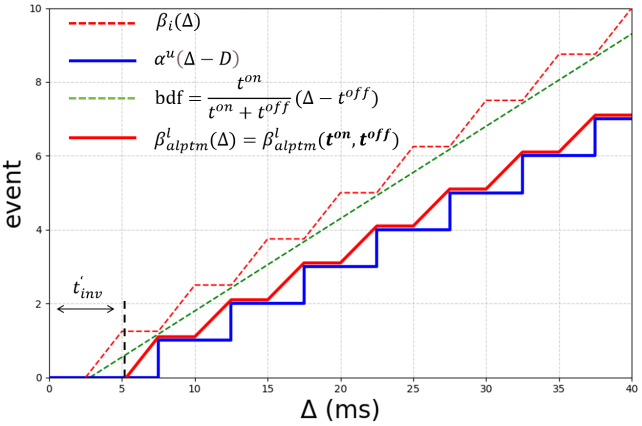


Figure 3: The service curve (in solid red) obtained by the PTM scheme specified by t_{off} and t_{on} set by the ALPTM, and the real-time constraint curve (in blue).

Problem Statement

In this study, we adopt PTM (Cheng et al. 2015) to manage the temperature. A PTM scheme uses parameters $\boldsymbol{t}^{\text{on}} = [t_1^{\text{on}}, t_2^{\text{on}}, ..., t_n^{\text{on}}]$ and $\boldsymbol{t}^{\text{off}} = [t_1^{\text{off}}, t_2^{\text{off}}, ..., t_n^{\text{off}}]$ to regulate the core's *active* and *sleep* periods within each cycle, with its configuration directly impacting service capacity, queue backlog, and temperature evolution.

Now we define our problem as follows:

Given an n-core pipelined system, WCETs c_i, and an end-to-end deadline D, the objective is to find online PTM parameters $\boldsymbol{t}^{on}$ and $\boldsymbol{t}^{off}$ that minimize the processor's peak temperature while satisfying both hard real-time and hardware feasibility constraints.

Motivation

As introduced in the Application Model above, the real-time requirements of the system are guaranteed if the service curve is no lower than the real-time constraint curve. To facilitate theoretical analysis, the method proposed in

(Cheng et al. 2021) employs a Bounded Delay Function (BDF) as an approximate lower bound of the service curve, thereby transforming the peak temperature optimization under hard real-time constraints into a linear programming problem. However, this method introduces a dual-layer approximation: first, it approximates the real-time constraint curve from above using a straight-line BDF; then, it reuses the same BDF as a lower bound to approximate the resulting service curve. As shown in Figure 2, these two approximations lead to an overly conservative strategy, where the service curves provided by all cores exceed the actual demand, indicating that further temperature reduction remains possible.

To achieve lower peak temperatures, it is necessary to reduce the approximation errors described above. This paper investigates whether the original nonlinear optimization problem can be addressed directly, without linear simplification. The complexity of this problem lies in the fact that each PTM decision influences future temperature trajectories, queue states, and service capacity, which collectively determine whether tasks can be completed on time. Traditional methods struggle to solve such problems efficiently, whereas deep reinforcement learning, leveraging the nonlinear representational capacity of neural networks, offers a viable alternative. By adopting this approach, a more precise PTM scheme can be learned, leading to two key improvements: first, the latency factor $t^{'}_{\text{inv}}$ can be extended without violating real-time constraints; second, the $active$ period $\boldsymbol{t}_{\text{on}}$ can be shortened, allowing the service curve to align more closely with the real-time constraint curve, as illustrated in Figure 3. Together, these contributions enable more effective reduction of peak temperature while still strictly satisfying all hard real-time requirements.

ALPTM

Building upon the system model and formal problem definition, this section presents the proposed ALPTM. ALPTM leverages deep reinforcement learning to generate PTM schemes that minimize peak temperature while guaranteeing hard real-time correctness. The agent adjusts PTM parameters online based on runtime feedback from temperature evolution, queueing behavior, and service capacity.

The state space, action space and reward function in ALPTM are formally defined as follows.

State Space

To ensure that the ALPTM agent perceives both thermal behavior and timing risks, the state representation must jointly capture temperature magnitude, thermal trends, mode-transition stability, and the system's proximity to real-time boundaries. Thus, the state space of our method includes the following characteristics:

- **Peak Temperature Feature:** Peak temperature directly corresponds to the global optimization objective. We include $\boldsymbol{T}^{\text{pt}} = \begin{bmatrix} T_1^{\text{pt}} & T_2^{\text{pt}} & \dots & T_n^{\text{pt}} \end{bmatrix}$, where T_i^{pt} denotes the maximum temperature of the i-th core within the previous sampling cycle.

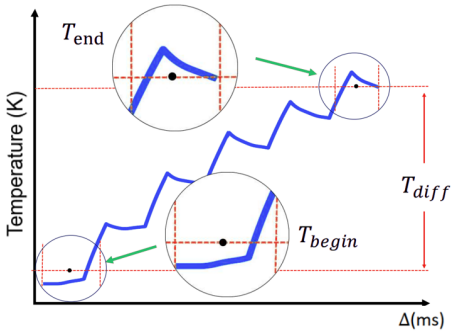


Figure 4: Processor temperature undergoes periodic changes over time.

- **Temperature Change Feature:** Thermal inertia implies that temperature differences across cycles offer predictive information regarding future thermal drift. As in Figure 4, we define $\boldsymbol{T}^{\text{diff}} = \boldsymbol{T}^{\text{end}} - \boldsymbol{T}^{\text{begin}} = \begin{bmatrix} T_1^{\text{diff}} & T_2^{\text{diff}} & \dots & T_n^{\text{diff}} \end{bmatrix}$, where $\boldsymbol{T}^{\text{begin}}$ and $\boldsymbol{T}^{\text{end}}$ denote average temperatures during the early and late portions of the preceding cycle. This feature captures fine-grained local temperature trends.
- **Mode Switching Frequency Feature:** Frequent transitions between $active$ and $sleep$ modes increase overhead and destabilize service capacity. Thus, we record $\boldsymbol{N}^{\text{prd}} = \begin{bmatrix} N_1^{\text{prd}} & N_2^{\text{prd}} & \dots & N_n^{\text{prd}} \end{bmatrix}$, representing the total mode switching counts of all cores in the previous cycle.
- **Latency Factor Feature:** Real-time guarantees depend on the relative positions of aggregated service and real-time constraint curves. To provide a scalar proxy for real-time slack, we compute $t^{\text{inv}} = \sum_{i=1}^{n} t_i^{\text{off}}$, which reflects how close the current PTM scheme is to violating the real-time boundary.

The state representation integrates all four feature groups:

$$\boldsymbol{S}_t = \left\{ \boldsymbol{T}^{\text{pt}}, \boldsymbol{T}^{\text{diff}}, \boldsymbol{N}^{\text{prd}}, t^{\text{inv}} \right\} \tag{10}$$

This combined representation allows the agent to reason about high dimensional coupled dynamics involving temperature, timing margins, and switching behavior, thereby improving policy stability and convergence.

Action Space

The PTM controller must determine, for each core, the durations of the $active$ and $sleep$ states within a cycle. Thus, the action produced by the policy network is defined as:

$$\boldsymbol{a}_t = \left\{ \boldsymbol{t}^{\text{on}}, \boldsymbol{t}^{\text{off}} \right\} \tag{11}$$

However, the raw Actor outputs must be mapped into a feasible action domain that respects hardware constraints and real-time requirements. Two post-processing steps are applied:

- **Action Range Enforcement:**

  The feasible domain of actions is constrained by hardware and real-time limitations. The lower bound t_{llm} is

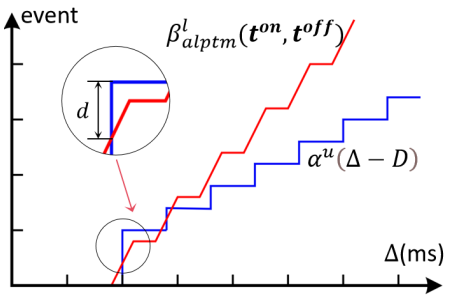


Figure 5: When $\beta^l_{\text{alpmt}}(\boldsymbol{t}^{\text{on}}, \boldsymbol{t}^{\text{off}}) < \alpha^u(\Delta - D)$, it indicates that the hard real-time constraint has been violated.

determined by mode switching times (t_i^{swon}, t_i^{swoff}). The upper bound t_{ulm} is constrained by the task's relative deadline D. To prevent excessive sleep time on a single core from overloading others, we set $t_{\text{ulm}} = 0.5 \times D$. This ensures real-time compliance while providing sufficient exploration space for the agent.

- **Integer-Cycle Alignment for RTC Analysis:** Real-time calculus requires the PTM cycle length to be integer-valued. We align the action as:$\boldsymbol{t}^{\text{off}} = \lceil \boldsymbol{t}^{\text{on}} + \boldsymbol{t}^{\text{off}} \rceil - \boldsymbol{t}^{\text{on}}$. As evident from the calculation formula, this process only amplifies $\boldsymbol{t}^{\text{off}}$, meaning the processed delay factor becomes larger. From Figure 2, it can be observed that if the processed aggregated service curve consistently does not exceed the real-time constraint curve, then the aggregated service curve generated by the original strategy also consistently does not exceed the real-time constraint curve. Importantly, the adjusted action is used only for service-curve evaluation and constraint checking; the original Actor output is used for gradient updates, preserving smooth policy learning.

Reward Function

The reward function is designed to guide the agent toward policies that reduce peak temperature, satisfy hard real-time constraints, maintain inter-core thermal balance, and operate near the real-time boundary where thermal savings are maximized. To achieve this, our reward function defined as bellow:

- **Temperature Optimization Reward:** The peak temperature serves as the core metric for optimization. To enhance sensitivity to temperature reduction, we employ an exponential function to characterize the temperature reward, providing agents with stronger positive incentives when lowering peak temperatures, as shown in Eq. (12). Additionally, considering the potential risks from local hotspots, we supplement the peak temperature with a mean value as a global smoothing metric. This prevents strategies from overloading individual cores while pursuing global optimization, as shown in Eq. (13). The combined temperature reward is expressed as Eq. (14).

$$r_1 = \begin{cases} e^{\frac{T_{\text{th}} - T_{\max}}{T_{\text{th}}}} - 1 & , \quad T_{\max} > T_{\text{th}} \\ -e^{\frac{T_{\max} - T_{\text{th}}}{T_{\text{th}}}} - 5 & , \quad T_{\max} < T_{\text{th}} \end{cases} \tag{12}$$

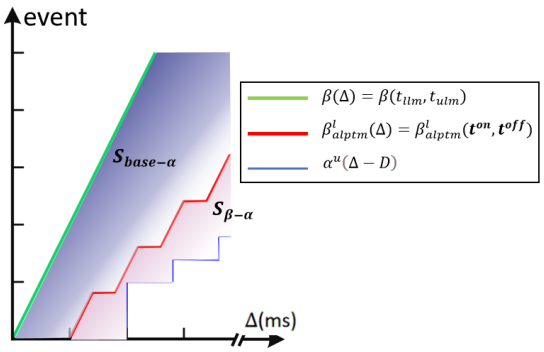


Figure 6: A diagram illustrating the integral value of curve spacing in the Limit-Approach Reward.

$$r_2 = \begin{cases} e^{\frac{T_{\text{th}} - T_\mu}{T_{\text{th}}}} - 1 & , \quad T_\mu > T_{\text{th}} \\ -e^{\frac{T_\mu - T_{\text{th}}}{T_{\text{th}}}} - 5 & , \quad T_\mu < T_{\text{th}} \end{cases} \tag{13}$$

$$R_1 = r_1 + r_2 \tag{14}$$

where $T_{\max}$ and T_μ denote the maximum and the average peak temperature of all cores within one sampling period, respectively, and $T_{\text{th}} = 373.15K$ is the theoretical maximum temperature threshold of the processor.

- **Real-time Constraint Violation Penalty:** Hard real-time guarantees must never be violated. When the aggregated service curve drops below the real-time constraint curve (Figure 5), we compute the vertical difference d at the point of violation and impose a penalty:

$$R_2 = \begin{cases} 0 & , \quad \beta^l_{\text{alptm}}(\Delta) \geq \alpha^u(\Delta - D) \\ -d & , \quad \beta^l_{\text{alptm}}(\Delta) < \alpha^u(\Delta - D) \end{cases} \tag{15}$$

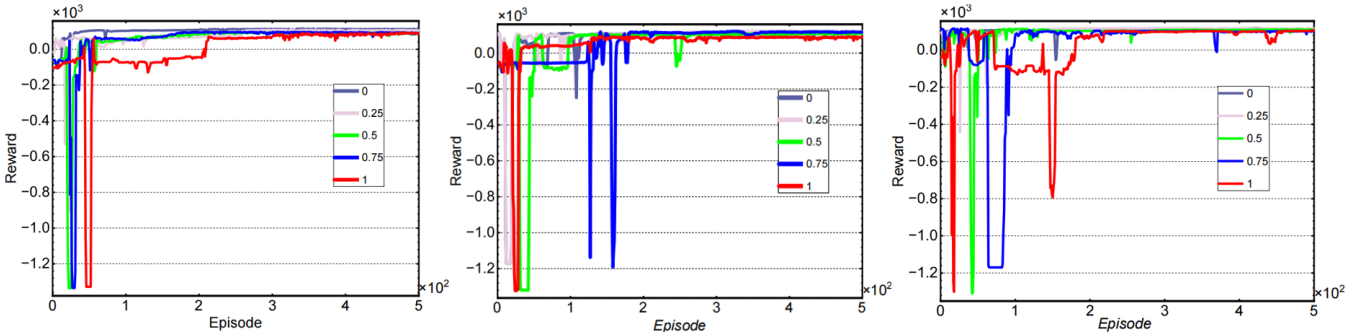
This term prevents the agent from trading correctness for temperature reduction.

- **Inter-Core Balance Reward:** Long-term thermal imbalance can cause local hotspots. To encourage thermal consistency, we include a balance metric combining the variance σ^2_{Temp} and temperature range Υ_{Temp}:

$$R_3 = max\left\{1, \frac{1}{\sigma^2_{\text{Temp}} + 0.5 * \Upsilon_{\text{Temp}}} - 1\right\} \tag{16}$$

This stabilizes temperature distribution across cores.

- **Limit-Approach Reward:** Recognizing that operating near real-time boundaries offers optimal low temperature performance but carries a high risk of constraint violations, we designed a limit-approximation reward term. This term encourages service behaviors that approach the boundaries without breaching them, allowing for precise control near the safety limits. The limit state is quantified by the area $S_{\beta-\alpha}$ between the two curves, as shown in Figure 6 and calculated by:



(a) H.263 (b) MP3 (c) MAD

Figure 7: Reward convergence curves during ALPTM training under different scenarios and jitter conditions.

Application	**WCETs**(ms)	**Period**(ms)
H.263	[1.32, 7.20, 5.40, 2.16]	50
MP3	[1.33, 5.60, 5.11, 3.57]	60
MAD	[2.40, 3.12, 3.60, 4.80]	50

Table 1: WCET and period of application.

Parameter	**Value**	**Parameter**	**Value**
Max epoch	500	Replay buffer size	1e5
Timesteps	300	Actor learning rate	1e-4
Batch size	256	Critic learning rate	1e-4
Critic architecture	$21\times128\times256\times256\times256\times1$		
Actor architecture	$13\times128\times256\times256\times256\times8$		

Table 2: ALPTM training hyperparameters.

$$S_{\beta-\alpha} = \int_0^t \beta^l_{\text{alptm}}(\Delta) - \alpha^u(\Delta - D)\mathrm{d}\Delta \tag{17}$$

Given that $S_{\beta-\alpha}$ can be very large, we normalize it using a baseline service curve $\beta(t_{\text{ulm}}, t_{\text{llm}})$, whose corresponding area is denoted as $S_{\text{base}-\alpha} = \int_0^t \beta(\Delta) - \alpha^u(\Delta - D)\mathrm{d}\Delta$. The limit-approximation reward is therefore defined as:

$$R_4 = \begin{cases} 1 - \frac{S_{\beta-\alpha}}{S_{\text{base}-\alpha}} & , \quad \beta^l_{\text{alptm}}(\Delta) \geq \alpha^u(\Delta - D) \\ 0 & , \quad \beta^l_{\text{alptm}}(\Delta) < \alpha^u(\Delta - D) \end{cases} \tag{18}$$

The overall reward integrates the four components with tunable weights k_1–$k_4 (k_1 = 1, k_2 = 0.25, k_3 = 0.25, k_4 = 1)$:

$$\text{Reward} = k_1 \cdot R_1 + k_2 \cdot R_2 + k_3 \cdot R_3 + k_4 \cdot R_4 \tag{19}$$

This composite formulation ensures that the policy converges toward safe, stable, and thermally optimal behaviors even under dynamic workloads and strong thermal inertia.

Case Study

This section evaluates the feasibility and effectiveness of the proposed ALPTM framework through systematic comparisons with four representative PTM-based approaches (Cheng et al. 2021), including: (1) FBGD: a greedy-descent PTM adjustment method based on the FBPT analysis framework ; (2) ANSA: a simulated annealing search algorithm operating in discrete PTM pattern space; (3) BS: a PBOO-based exhaustive pattern enumeration method; (4) SDP: a heuristic sub-deadline–partitioning strategy. These baselines cover heuristic, stochastic search, and static pattern–construction paradigms, enabling a comprehensive evaluation of reinforcement learning in continuous action spaces.

Experiments adopt the classical timing model $PJD(p, j, d)$ to generate event arrivals. Peak temperature optimization is performed on three commonly used streaming applications: H.263, MP3, and MAD, whose WCET vectors $\boldsymbol{c}$ and periods are listed in Table 1 (Oh and Ha 2002). To evaluate adaptability under different levels of burstiness, we introduce a jitter factor ξ, defining jitter as: $j = \xi * p$.

All the experiments were conducted on the ARM 4-core simulation platform implemented by HotSpot toolbox (Huang et al. 2006), and the parameter configurations of each method were exactly the same. The simulation duration is 60 s, the sampling interval is 300 ms, and all mode-switching delays are fixed at $t_i^{\text{swon}} = t_i^{\text{swoff}} = 1$ ms.

Convergence Behavior

Figure 7 presents the reward convergence curves for the three flow applications during ALPTM training, evaluated under different levels of jitter. To ensure comparability across all runs, identical hyperparameters were employed, as summarized in Table 2.

Across all settings, the reward exhibits a monotonic upward trend with training iterations and eventually stabilizes, indicating that ALPTM successfully captures the coupled relationship among cross-cycle queueing dynamics, service capacity variation, and temperature evolution. Under light jitter, state-transition volatility is low, allowing the agent to converge more rapidly because exploration can more con-

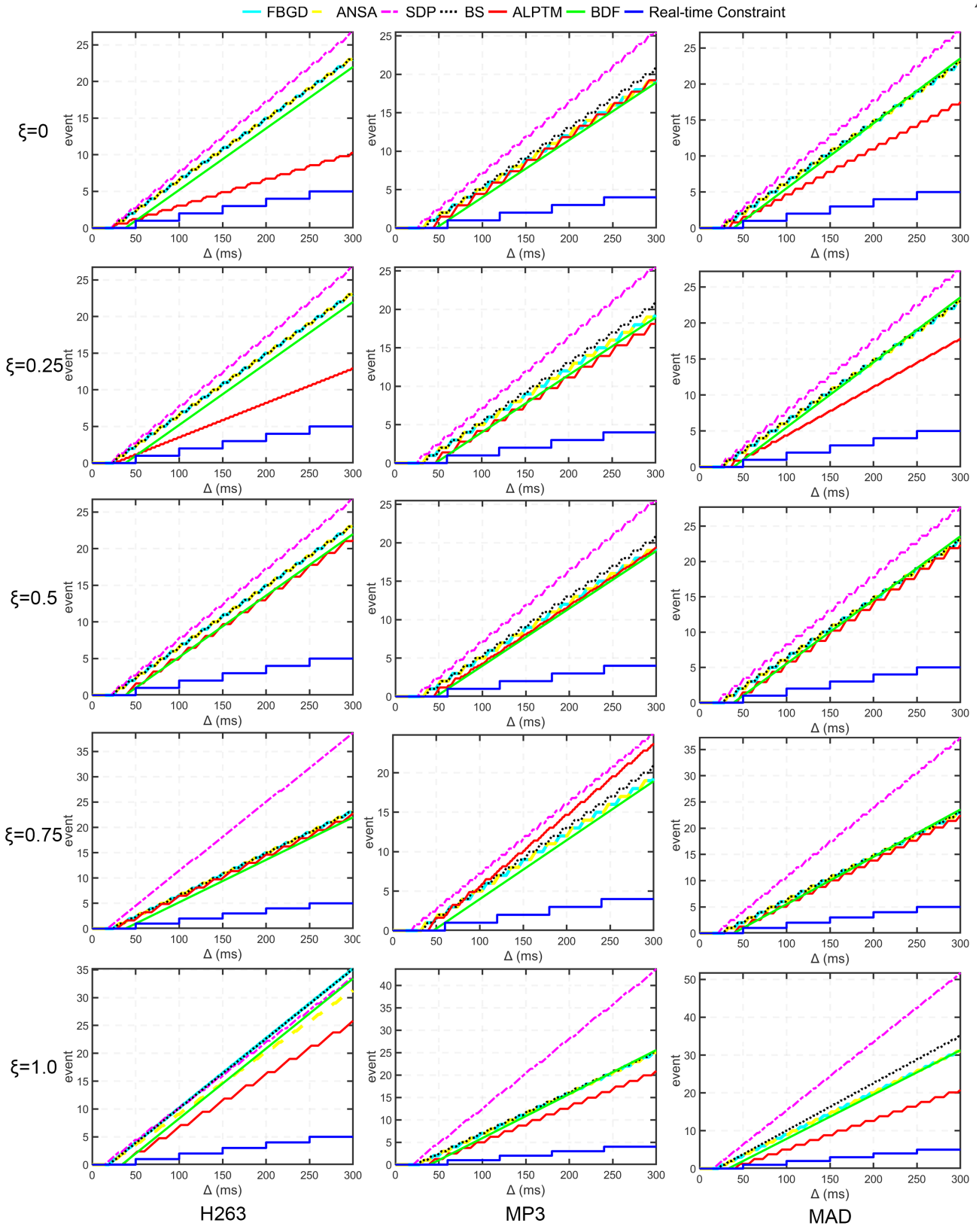


Figure 8: Real-time constraint curves versus service curves under PTM strategies formulated by various methods across different scenarios and jitter conditions.

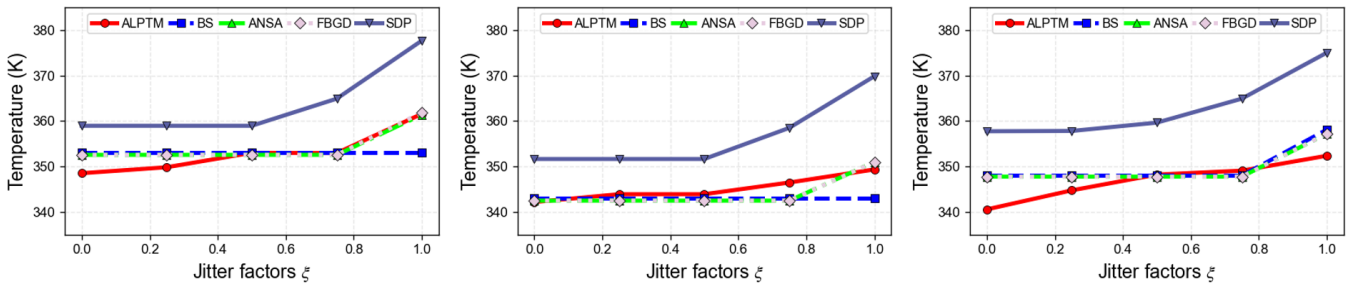


Figure 9: A comparison of peak temperatures under strategies formulated by various methods across different scenarios and jitter conditions.

sistently focus on high-quality action regions. Under heavy jitter, state perturbations significantly increase due to abrupt fluctuations in arrival patterns. Consequently, policy convergence becomes slower. Nevertheless, ALPTM consistently reaches a stable reward plateau, demonstrating that the reinforcement learning framework maintains robustness despite high non-determinism and strong thermal inertia. This validates the stability of the learned policy across diverse runtime environments.

Verification of Real-time Constraints

We evaluate whether the PTM schemes generated by ALPTM satisfy hard real-time constraints. Figure 8 compares the real-time constraint curves against the service curves produced by different algorithms. This comparison spans the three streaming applications under various jitter conditions, with the BDF curve included as a benchmark.

Across all tested configurations, the service curve consistently remains above the real-time constraint curve, with no boundary violations observed. This means that all of them have met the real-time constraints. By comparing the positional relationship between the service curves and the BDF curve corresponding to different algorithms, a notable pattern can be observed: in most cases, the service curve corresponding to the APLTM algorithm proposed in this paper has a smaller slope than the BDF curve and lies closer to the real-time constraint curve; whereas the service curves corresponding to the other four comparative methods exhibit a greater slope than the BDF curve and are positioned farther from the real-time constraint curve. This indicates that the comparative methods are more conservative than our approach, which in turn leads to a higher peak processor temperature. This occurs because a conservative strategy primarily extends the $active$ periods of the cores to maintain a larger safety margin for real-time compliance. The resulting increase in $active$ mode operation raises power dissipation, which in turn leads to a higher peak temperature.

Comparison of peak temperatures

We evaluate the peak temperature performance of ALPTM against four comparative methods under varying jitter conditions. As shown in Figure 9, the resulting peak temperatures are reported for different streaming applications.

The experimental results demonstrate that our method achieves the best or closely competitive performance in almost cases. As can be observed from the figure, when the jitter is small, our method generally achieves the lowest peak temperature, outperforming other methods by up to 7–8 K (e.g., in MAD at $\xi = 0$). This is directly reflected in the service-curve comparisons in Figure 8, where our method yields a curve that most closely follows the real-time constraint. In essence, by solving the original nonlinear optimization, our approach learns to shorten the $active$ period $\boldsymbol{t}_{\text{on}}$ while still guaranteeing the required service. This reduces the time the core spends idling at high power while waiting for execution, thereby lowering the peak temperature. Even under larger jitter conditions, our method maintains comparable or better performance, demonstrating its stronger adaptability to dynamic system states.

Conclusion

This paper presented ALPTM, an autonomous learning framework for Periodic Thermal Management in hard real-time pipeline systems. Motivated by the inherent nonlinear coupling among queue dynamics, service capacity, and thermal evolution, we formalized the PTM optimization as a continuous-action Markov Decision Process and employed the TD3 algorithm to learn adaptive $sleep$/$active$ schemes. By directly solving the underlying nonlinear optimization problem, ALPTM departs from prior linear approximation methods. It dynamically adjusts decisions using runtime feedback, enabling precise control that effectively handles thermal inertia and stochastic workload variations. Experimental evaluations across different streaming applications and workloads confirm that ALPTM maintains strict compliance with end-to-end deadlines while consistently matching or lowering peak temperatures compared to baselines.

Overall, these results reinforce the value of reinforcement learning as a practical and adaptive tool for real-time thermal control in systems with nonlinear constraints. Future work will explore cross-application generalization to move toward fully autonomous system-level thermal management.

Acknowledgments

This work has been supported in part by the National Natural Science Foundation of China (Grant 61902442), in part

by the Basic Public Welfare Research Program of ZheJiang Province (Grant LTGS24F020002).

References

Benini, L.; Bogliolo, A.; and De Micheli, G. 2000. A survey of design techniques for system-level dynamic power management. *IEEE transactions on very large scale integration (VLSI) systems*, 8(3): 299–316.

Chantem, T.; Dick, R. P.; and Hu, X. S. 2008. Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs. In *Proceedings of the conference on Design, automation and test in Europe*, 288–293.

Cheng, L.; Huang, K.; Chen, G.; Hu, B.; and Knoll, A. 2015. Periodic thermal management for hard real-time systems. In *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 1–10. IEEE.

Cheng, L.; Huang, K.; Chen, G.; Hu, B.; and Knoll, A. 2016. Minimizing peak temperature for pipelined hard real-time systems. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1090–1095.

Cheng, L.; Huang, K.; Mi, L.; Chen, G.; Knoll, A.; and Zhang, X. 2021. Peak temperature analysis and optimization for pipelined hard real-time systems. *Information Sciences*, 575: 666–697.

Dey, S.; Isuwa, S.; Saha, S.; Singh, A. K.; and McDonald-Maier, K. 2022. CPU-GPU-memory DVFS for power-efficient MPSoC in mobile cyber physical systems. *Future Internet*, 14(3): 91.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Gill, S. S.; Tuli, S.; Toosi, A. N.; Cuadrado, F.; Garraghan, P.; Bahsoon, R.; Lutfiyya, H.; Sakellariou, R.; Rana, O.; Dustdar, S.; et al. 2020. ThermoSim: Deep learning based framework for modeling and simulation of thermal-aware resource management for cloud computing environments. *Journal of Systems and Software*, 166: 110596.

Hettiarachchi, P. M.; Fisher, N.; et al. 2013. Achieving thermal-resiliency for multicore hard-real-time systems. In *2013 25th Euromicro Conference on Real-Time Systems*, 37–46. IEEE.

Huang, K.; Chen, G.; Buckl, C.; and Knoll, A. 2012. Conforming the runtime inputs for hard real-time embedded systems. In *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, 430–436. New York, NY, USA: Association for Computing Machinery. ISBN 9781450311991.

Huang, K.; Santinelli, L.; Chen, J.-J.; Thiele, L.; and Buttazzo, G. C. 2009. Adaptive Dynamic Power Management for Hard Real-Time Systems. In *2009 30th IEEE Real-Time Systems Symposium*, 23–32.

Huang, W.; Ghosh, S.; Velusamy, S.; Sankaranarayanan, K.; Skadron, K.; and Stan, M. 2006. HotSpot: a compact thermal modeling methodology for early-stage VLSI design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5): 501–513.

Ilager, S.; Ramamohanarao, K.; and Buyya, R. 2020. Thermal prediction for efficient energy management of clouds using machine learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(5): 1044–1056.

Khan, H.; Ali, A.; Alshmrany, S.; et al. 2023. Energy-Efficient Scheduling Based on Task Migration Policy Using DPM for Homogeneous MPSoCs. *Computers, Materials & Continua*, 75(1).

Mohaqeqi, M.; Kargahi, M.; and Fouladi, K. 2016. Stochastic thermal control of a multicore real-time system. In *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 208–215. IEEE.

Mulas, F.; Atienza, D.; Acquaviva, A.; Carta, S.; Benini, L.; and De Micheli, G. 2009. Thermal balancing policy for multiprocessor stream computing platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(12): 1870–1882.

Narang, G.; Deshwal, A.; Ayoub, R.; Kishinevsky, M.; Rao Doppa, J.; and Pande, P. P. 2023. Dynamic power management in large manycore systems: A learning-to-search framework. *ACM Transactions on Design Automation of Electronic Systems*, 28(5): 1–21.

Oh, H.; and Ha, S. 2002. Hardware-software cosynthesis of multi-mode multi-task embedded systems with real-time constraints. In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign*, CODES '02, 133–138. New York, NY, USA: Association for Computing Machinery. ISBN 1581135424.

ul Islam, F. M. M.; Lin, M.; Yang, L. T.; and Choo, K.-K. R. 2018. Task aware hybrid DVFS for multi-core real-time systems using machine learning. *Information Sciences*, 433: 315–332.

Yao, Y. 2023. Game-of-life temperature-aware DVFS strategy for tile-based chip many-core processors. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 13(1): 58–72.

Yun, B.; Shin, K. G.; and Wang, S. 2011. Thermal-aware scheduling of critical applications using job migration and power-gating on multi-core chips. In *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, 1083–1090. IEEE.

Zhou, J.; Li, L.; Vajdi, A.; Zhou, X.; and Wu, Z. 2021. Temperature-constrained reliability optimization of industrial cyber-physical systems using machine learning and feedback control. *IEEE Transactions on Automation Science and Engineering*, 20(1): 20–31.