Accelerated Optimal Task Phasing for LET-Based Cause–Effect Chains

Jongwon Lee, Namcheol Lee, Saehwa Kim, *Member, IEEE*, and Seongsoo Hong, *Senior Member, IEEE*

***Abstract*—Task phases are known to affect the end-to-end (E2E) latency of Logical Execution Time (LET)-based cause–effect chains (CECs). However, existing methods for computing optimal phases that minimize E2E latency either incur substantial search costs or rely on harmonicity assumptions. To address these limitations, we propose an exact method that substantially reduces search cost relative to an existing enumeration-based method without relying on harmonicity assumptions. Using a zero-wait alignment principle, we derive a finite, optimality-preserving set of candidate phases for a task that extends a subchain of the CEC with fixed task phases. We then develop a depth-first branch-and-bound algorithm that explores the search tree formed by recursively extending the subchain one task at a time using these candidates, starting from a maximum-period task. The algorithm incorporates singleton acceleration that computes the unique candidate phase and latency increment in closed form under a divisibility condition, and lower-bound pruning that discards branches that cannot improve the best solution found so far. Across all three synthetic workloads with 500 five-task CECs each, the proposed algorithm achieves speedups of 255 to $7.32 \cdot 10^7$ over the enumeration-based method, measured as ratios of their 99th-percentile execution times.**

***Index Terms*—Cause–effect chains, design optimization, end-to-end latency, logical execution time, real-time systems, task phases.**

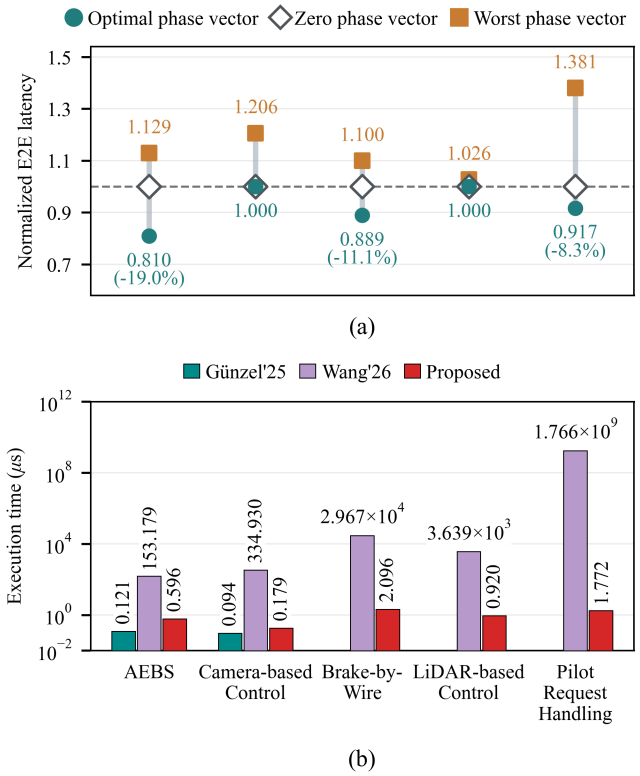

Fig. 1. Five application-derived CECs: (a) E2E latency under optimal, zero, and worst phase vectors, normalized to that under the zero phase vector (synchronous release with all task phases zero). Latencies are computed over a 1 ms integer phase grid under the standard LET model with implicit deadlines. (b) Execution time to compute an optimal phase vector (experimental setup in Section VI). Task periods in CEC order (ms): autonomous emergency braking system (AEBS) [7] $(10, 50, 10, 50)$; camera-based control [9] $(1000, 2000, 40)$; Brake-by-Wire [10], [11] $(20, 30, 40, 50, 60)$; LiDAR-based control [5], [11] $(33, 400, 15, 15, 5)$; pilot request handling [12] $(40, 60, 100, 100, 100, 60, 40)$. Günzel'25 is applicable only to AEBS and camera-based control. The compared methods are summarized in Table I.

I. Introduction

MANY safety-critical embedded functions in industry are implemented as cause–effect chains (CECs), sequences of multi-rate real-time tasks that propagate data from sensing to actuation [1]–[5]. The timing behavior of a CEC is characterized by its end-to-end (E2E) latency, for which two metrics are usually considered: the maximum reaction time (MRT), the maximum time required for an external activity to be processed, and the maximum data age (MDA), the maximum age of the data used in an actuation [6], [7]. MRT and MDA are equivalent for CECs of periodic tasks under logical execution time (LET) semantics [8]. Since we consider such CECs, we treat MRT and MDA as a single metric and refer to it as *E2E latency*.

LET semantics has received considerable attention for its deterministic timing behavior and robustness to functional extensions and changes [13]–[15]. Under LET semantics, each job of a task reads its input and writes its output at fixed logical instants independent of when the job actually executes; the interval between them is called the *LET interval* [14]. We call CECs whose tasks follow these semantics *LET-based CECs*. In the standard LET model, each job's LET interval extends from its release time to its deadline; the timing of these communication events is then determined solely by the task's period, deadline, and phase, where phase denotes the task's initial release time and is also referred to as an offset [7], [16]. We call the tuple of the phases of all tasks in a CEC its *phase vector*. Since a phase vector determines how the write-

This work was supported by Hankuk University of Foreign Studies Research Fund. *(Corresponding author: Saehwa Kim.)*

J. Lee is with the Department of Future Automotive Mobility, Seoul National University, Seoul, Republic of Korea (e-mail: jwlee@redwood.snu.ac.kr).

N. Lee and S. Hong are with the Department of Electrical and Computer Engineering, Seoul National University, Seoul, Republic of Korea (e-mail: nlee@redwood.snu.ac.kr; sshong@redwood.snu.ac.kr).

S. Kim is with the Department of AI Robotics, Hankuk University of Foreign Studies, Yongin, Republic of Korea (e-mail: ksaehwa@hufs.ac.kr).

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

events of each task align with the read-events of its successor, it directly governs the E2E latency of the CEC. Accordingly, we call a phase vector optimal if it minimizes the E2E latency over all phase vectors of the CEC. Figure 1(a) illustrates how phase vectors affect E2E latency in five application-derived CECs from automotive, robotic, and avionics systems.

The E2E latency may need to meet a specified upper bound, referred to as an *E2E latency requirement* [6], [7]. Beyond reducing latency, computing the minimum E2E latency determines whether this requirement can be met by changing the phase vector alone or whether other configuration parameters, such as task periods, must also be changed. Since designers may repeat this assessment for multiple configurations, fast computation helps them explore design alternatives efficiently.

Prior studies have proposed methods for computing optimal phase vectors and their corresponding E2E latencies [7], [15], [16]. However, these methods either incur substantial combinatorial search cost [15], [16] or provide closed-form solutions under specific semi-harmonic period conditions [7]. Table I compares our method with state-of-the-art methods. For the seven-task pilot request handling CEC in Fig. 1(b), for example, the enumeration-based method [15] takes about 29.4 min ($1.766 \cdot 10^9$ μs) to compute an optimal phase vector, while the closed-form method [7] is inapplicable.

To address these limitations, we propose a method that computes an optimal phase vector and its corresponding E2E latency without harmonicity assumptions and substantially reduces search cost relative to the enumeration-based method. For the pilot request handling CEC, the proposed method obtains the same minimum E2E latency in 1.772 μs (Fig. 1(b)). The main contributions are as follows.

- Using a zero-wait alignment principle, we derive a finite, optimality-preserving set of candidate phases for a task that extends a subchain of the CEC with fixed task phases (Theorem 1).
- We develop a depth-first branch-and-bound algorithm that finds an optimal phase vector and its corresponding E2E latency by recursively extending the subchain one task at a time using these candidates, starting from a maximum-period task (Theorem 3). The search is accelerated by singleton acceleration and lower-bound pruning.
- We derive closed-form lower and upper bounds on the minimum E2E latency (Theorem 2).
- We prove the optimality of the zero phase vector for CECs satisfying a divisibility condition and derive their minimum E2E latencies in closed form (Corollary 4). Two-task CECs always satisfy this condition.

Across all three synthetic workloads with 500 five-task CECs each, the proposed algorithm achieves speedups of 255 to $7.32 \cdot 10^7$ over the enumeration-based method, measured as ratios of their 99th-percentile execution times.

The remainder of this article is organized as follows. Section II reviews related work. Section III presents the system model and problem formulation. Section IV derives the candidate phase sets and latency bounds. Section V presents the search algorithm and establishes its correctness. Section VI evaluates the proposed method, and Section VII concludes the article.

TABLE I
COMPARISON OF METHODS FOR COMPUTING OPTIMAL PHASE VECTORS FOR LET-BASED CECS

Criterion	Günzel and Becker [7]	Wang et al. [15]	Proposed
Period assumption	Semi-harmonic[a]	None	None
Procedure	Closed form	Enumeration	Branch-and-bound[b]
Cost	Low	High	Low[c]

[a] Max-harmonic or $(2, k)$-max-harmonic period sets, detailed in Section VI-B.
[b] Branches over reduced sets and prunes branches using a lower bound, as detailed in Sections IV and V.
[c] Low is relative to the enumeration-based method [15]; the worst-case complexity remains combinatorial.

II. RELATED WORK

When discussing prior work on CECs, we use *latency* as a collective term for the E2E latency metrics considered in the respective studies. We distinguish two related problems concerning the latency of CECs: (1) latency analysis, which characterizes latency for a given system configuration, and (2) configuration design, which determines system configuration parameters to meet latency requirements or reduce latency. Our method addresses the configuration design problem.

For the analysis problem, latency analysis frameworks have been developed for CECs in automotive and distributed systems [17]–[19]. Communication semantics determine the producer–consumer job relationships and can lead to different latencies [20], [21]. Günzel et al. [8] established the equivalence of MRT and MDA using partitioned job chains under general assumptions applicable to LET-based CECs. For LET-based CECs, Günzel, Becker, and Casini [11] developed a shape-aware analysis that characterizes the reaction-time curve and enables the computation of maximum and average latency. These studies provide a basis for latency analysis of CECs but do not provide a method for computing an optimal phase vector. Our method uses the equivalence result of Günzel et al. [8] and adopts their partitioned-job-chain representation of latency.

Configuration design studies can be distinguished by whether the CECs considered use LET semantics. Among studies of CECs that do not use LET semantics, several determine task configurations or synthesize job-level dependencies [22]–[25]. Ali Ahmad et al. [26] propose a heuristic that determines task phases jointly with runnable-to-task mappings and within-task execution orders. However, this heuristic does not guarantee optimality. Unlike these studies, our method computes an optimal phase vector under the standard LET model.

For LET-based CECs, two approaches have been studied. One reconfigures LET intervals, whereas the other determines task phases with prescribed LET intervals. The first approach reconfigures LET intervals beyond the standard LET model, using virtual offsets and virtual deadlines [9], worst-case response times under optimized per-core priorities [27], or precedence constraints between jobs [28], [29]. Reconfiguring LET intervals can reduce latency beyond what task phasing alone achieves under the standard LET model, but requires

TABLE II
KEY NOTATION USED IN SECTION III

Symbol	Description
$\Psi_{1:N}$	CEC $(\tau_1 \to \cdots \to \tau_N)$ consisting of $N \geq 2$ ordered tasks
$\Psi_{a:b}$	Contiguous task subchain $(\tau_a \to \cdots \to \tau_b)$ of $\Psi_{1:N}$, where $1 \leq a \leq b \leq N$
τ_i	i-th task in $\Psi_{1:N}$, with period T_i and phase ϕ_i
τ_i^k	k-th job of τ_i
$\widehat{p}$	Smallest index of a maximum-period task in $\Psi_{1:N}$
$\vec{\varphi}_{a:b}$	Phase vector $(\phi_a, \ldots, \phi_b)$ of $\Psi_{a:b}$
$H_{a:b}$	Hyperperiod of $\Psi_{a:b}$
$M_{a:b}$	Number of $\tau_{\widehat{p}}$-partitioned job chains of $\Psi_{a:b}$ per $H_{a:b}$
$\psi_{a:b}^{(j)}$	j-th $\tau_{\widehat{p}}$-partitioned job chain of $\Psi_{a:b}$, $0 \leq j < M_{a:b}$
$s_{a:b}^{(j)}$	Read-event time of the first job of $\psi_{a:b}^{(j)}$
$w_{a:b}^{(j)}$	Write-event time of the last job of $\psi_{a:b}^{(j)}$
$\mathcal{S}_{a:b}$	Tuple of $s_{a:b}^{(j)}$ for $0 \leq j < M_{a:b}$
$\mathcal{W}_{a:b}$	Tuple of $w_{a:b}^{(j)}$ for $0 \leq j < M_{a:b}$
$\mathrm{Lat}(\Psi_{a:b}; \vec{\varphi}_{a:b})$	E2E latency of $\Psi_{a:b}$ under $\vec{\varphi}_{a:b}$

ensuring feasibility. Our method complements this approach. It determines whether a latency requirement can be met without reconfiguring LET intervals and can also be applied to prescribed LET intervals obtained through reconfiguration, provided that each task's logical read- and write-events remain periodic with its task period (Section III-A).

The second approach determines task phases alone, as they are the only parameters affecting latency when task periods and LET intervals are prescribed. Martinez et al. [16] propose a depth-parameterized phase-assignment procedure. Restricted-depth search is heuristic, whereas full-depth search exhaustively explores non-equivalent integer phase assignments. As summarized in Table I, Günzel and Becker [7] derive optimal phase vectors in closed form at low cost, but only for specific semi-harmonic period sets. Wang et al. [15] introduce the zeroLET model to simplify the analysis of CECs with prescribed LET intervals and derive the maximum E2E latency over all integer phase vectors in closed form. Like the full-depth search of Martinez et al. [16], they enumerate non-equivalent integer phase vectors, but in the zeroLET model, and recover an optimal phase vector and the minimum E2E latency for the original CEC. Our method requires no harmonicity assumptions and reduces computation cost relative to enumeration-based methods [15], [16] by searching derived sets of candidate phases with singleton acceleration and lower-bound pruning, while preserving optimality. Its worst-case complexity, however, remains combinatorial.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. Workload Model and LET Semantics

The system model in Section III follows prior studies of CECs [7], [9], [15], [16], [21], with key notation summarized in Table II. The proposed analysis is applicable to both single-core and multicore systems.

We focus on a CEC $\Psi_{1:N} = (\tau_1 \to \cdots \to \tau_N)$ with $N \geq 2$. For $1 \leq a \leq b \leq N$, let $\Psi_{a:b} = (\tau_a \to \cdots \to \tau_b)$ denote the contiguous task subchain from τ_a to τ_b. Each task τ_i is characterized by a tuple (T_i, ϕ_i), where T_i and ϕ_i denote the period and phase of τ_i, respectively, with $T_i \in \mathbb{N}_{>0}$ and $\phi_i \in \mathbb{R}$. We omit the worst-case execution time from this tuple because execution times do not affect E2E latency under LET semantics. Let τ_i^k denote the k-th job of τ_i. Assuming implicit deadlines, τ_i^k has release time $r_i^k = \phi_i + k \cdot T_i$ and deadline $\phi_i + (k+1) \cdot T_i$, so that $\phi_i = r_i^0$. To model steady-state behavior, we allow negative job indices and let $k \in \mathbb{Z}$; for example, τ_i^{-1} is released one period before τ_i^0 [9], [21].

Under the standard LET model, in which each job reads its input at its release time and writes its output at its deadline, the read- and write-events of τ_i^k occur at $\mathrm{re}(\tau_i^k) = r_i^k$ and $\mathrm{we}(\tau_i^k) = r_i^{k+1}$, respectively. When a producer write-event and successor read-event occur at the same logical time, the written value is available to the successor. A successor task reads the latest data written to the shared resource by its immediate predecessor, with τ_1 reading external input and τ_N writing external output.

Setting the LET interval length of each task equal to its period entails no loss of generality in determining an optimal phase vector. For prescribed LET interval lengths $\lambda_i \geq 0$, each job of τ_i reads at its release and writes λ_i time units later. By the zeroLET decomposition of Wang et al. [15, Theorem 1], shifting each phase ϕ_i of a phase vector in the setting with LET interval lengths T_i by $-\sum_{j=1}^{i-1}(T_j - \lambda_j)$ yields a phase vector for LET interval lengths λ_i that preserves write-to-read time differences between adjacent tasks and whose E2E latency differs from that of the original vector by the constant $\sum_{i=1}^{N}(\lambda_i - T_i)$. Invertibility and this constant latency difference preserve optimality.

Following prior phasing studies of LET-based CECs [7], [15], [16], we focus on phase optimization of CECs, assuming that the prescribed LET intervals can be realized by a schedulable task set. The analysis does not depend on a specific scheduling policy and applies under fixed- or dynamic-priority schedulers. For example, for independent periodic tasks under preemptive rate-monotonic (RM) scheduling on a single processor, without blocking or self-suspension, schedulability under synchronous release suffices for all phase assignments. In general, the task set must be verified to be schedulable under the selected scheduling model, including the effects of phase changes, using an appropriate schedulability test.

B. Job Chains and End-to-End Latency

For adjacent tasks τ_i and τ_{i+1}, two jobs τ_i^u and τ_{i+1}^v are *connectable* if $\mathrm{we}(\tau_i^u) \leq \mathrm{re}(\tau_{i+1}^v)$. Building on this relation, to formalize the unified MRT/MDA metric introduced in Section I, we use three established job-chain concepts from prior work [7]. First, for $\Psi_{a:b}$, the m-th *immediate forward job chain* starts from τ_a^m and connects to the earliest connectable job of each subsequent task, terminating at τ_b. Second, the m-th *immediate backward job chain* ends at τ_b^m and traces back to the latest connectable job of each predecessor task until reaching τ_a. Finally, combining the two, for $p \in \{a, \ldots, b\}$, the m-th *τ_p-partitioned job chain*, denoted $pc_p^m(\Psi_{a:b})$, merges the immediate backward job chain of $\Psi_{a:p}$ ending at τ_p^m with the immediate forward job chain of $\Psi_{p:b}$ starting at τ_p^{m+1}; the backward and forward parts degenerate to a single job when $p = a$ and $p = b$, respectively [7], [8]. If $pc_p^m(\Psi_{a:b})$ starts at τ_a^u and ends at τ_b^v, its length is $\ell(pc_p^m(\Psi_{a:b})) = \mathrm{we}(\tau_b^v) - \mathrm{re}(\tau_a^u)$.

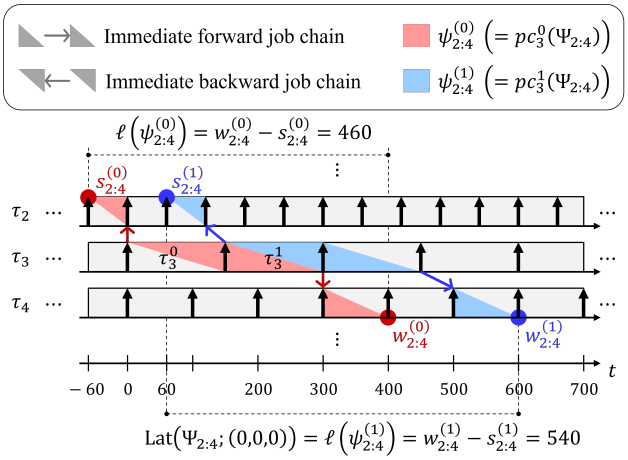

Fig. 2. The two τ_3-partitioned job chains $\psi^{(0)}_{2:4}$ and $\psi^{(1)}_{2:4}$ of the LET-based task subchain $\Psi_{2:4} = (\tau_2 \to \tau_3 \to \tau_4)$, with $\tau_2 = (60, 0)$, $\tau_3 = (150, 0)$, and $\tau_4 = (100, 0)$, for which $H_{2:4} = 300$, $\widehat{p} = 3$, and $M_{2:4} = 2$. For $j \in \{0, 1\}$, $\psi^{(j)}_{2:4}$ combines the immediate backward job chain ending at τ^j_3 with the immediate forward job chain starting at τ^{j+1}_3. Their first-read times are $\mathcal{S}_{2:4} = (-60, 60)$, their last-write times are $\mathcal{W}_{2:4} = (400, 600)$, and their lengths are $\ell(\psi^{(j)}_{2:4}) = w^{(j)}_{2:4} - s^{(j)}_{2:4}$, yielding 460 and 540, respectively.

A partitioned job chain is a logical construct for analyzing E2E latency and does not represent an actual data-propagation path.

Fig. 2 illustrates the two τ_3-partitioned job chains of $\Psi_{2:4}$ and its E2E latency. Let $\widehat{p} = \min \arg\max_{1 \le i \le N} T_i$. We assume that every task subchain $\Psi_{a:b}$ considered below contains $\tau_{\widehat{p}}$. For each such task subchain, let $\vec{\varphi}_{a:b} = (\phi_a, \ldots, \phi_b)$, $H_{a:b} = \mathrm{lcm}(T_a, \ldots, T_b)$, and $M_{a:b} = H_{a:b}/T_{\widehat{p}}$. In particular, $M_{1:N} = H_{1:N}/T_{\widehat{p}}$ is the ratio of the hyperperiod of the entire CEC to its maximum period, equal to the number of $\tau_{\widehat{p}}$-partitioned job chains of $\Psi_{1:N}$ within one hyperperiod $H_{1:N}$. By the $H_{a:b}$-periodicity of $\Psi_{a:b}$, every $\tau_{\widehat{p}}$-partitioned job chain is a time-shifted copy of one of the $M_{a:b}$ job chains, shifted by an integer multiple of $H_{a:b}$. We denote these job chains by $\psi^{(j)}_{a:b} = pc^j_{\widehat{p}}(\Psi_{a:b})$ for $0 \le j < M_{a:b}$. Let $s^{(j)}_{a:b}$ and $w^{(j)}_{a:b}$ denote the first-read and last-write times of $\psi^{(j)}_{a:b}$, respectively, i.e., the read-event time of its first job and the write-event time of its last job. Then, $\ell(\psi^{(j)}_{a:b}) = w^{(j)}_{a:b} - s^{(j)}_{a:b}$. Furthermore, let $\mathcal{S}_{a:b} = (s^{(0)}_{a:b}, \ldots, s^{(M_{a:b}-1)}_{a:b})$ and $\mathcal{W}_{a:b} = (w^{(0)}_{a:b}, \ldots, w^{(M_{a:b}-1)}_{a:b})$. Prior work established that the maximum length among all $\tau_{\widehat{p}}$-partitioned job chains of $\Psi_{1:N}$ equals the E2E latency [7], [8]. Applying this result to each task subchain $\Psi_{a:b}$, we define the E2E latency of $\Psi_{a:b}$ as

$$\mathrm{Lat}(\Psi_{a:b}; \vec{\varphi}_{a:b}) = \max_{0 \le j < M_{a:b}} \left(w^{(j)}_{a:b} - s^{(j)}_{a:b}\right). \tag{1}$$

Choosing $\widehat{p}$ as above minimizes $M_{a:b}$ in Eq. (1).

C. Problem Formulation

For a given CEC $\Psi_{1:N}$, the periods are fixed; only task phases are optimization variables. Since a common shift of all task phases preserves relative event times and hence every job-chain length, we may set $\phi_{\widehat{p}} = 0$ without loss of generality. We therefore define the phase-vector domain $\Theta = \{\vec{\varphi}_{1:N} \in \mathbb{R}^N \mid \phi_{\widehat{p}} = 0\}$. Restricting the search to Θ loses no optimal solution. Corollary 3 later establishes that the minimum over

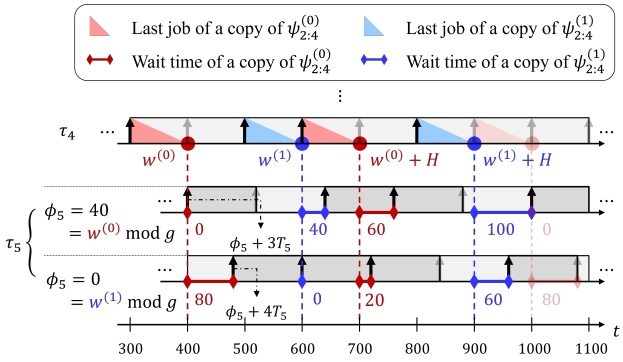

Fig. 3. Candidate phases for the right extension of $\Psi_{2:4}$ by $\tau_5 = (120, \phi_5)$, with the phases of $\Psi_{2:4}$ fixed as in Fig. 2. The phases $\phi_5 = 40$ and $\phi_5 = 0$ cause the earliest connectable jobs of τ_5 to read at the last-write times of $\psi^{(0)}$ and $\psi^{(1)}$, respectively, yielding zero wait time. Here, $\psi^{(j)}$ denotes $\psi^{(j)}_{2:4}$ for $j \in \{0, 1\}$, and H denotes $H_{2:4} = 300$.

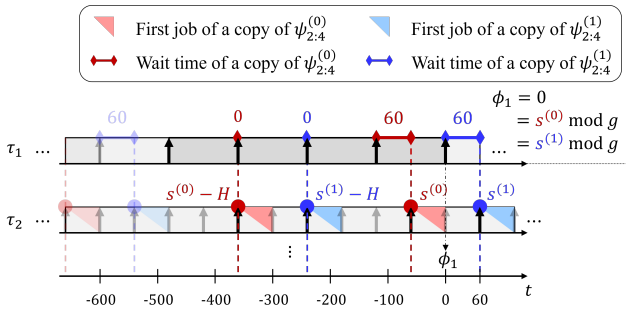

Fig. 4. Candidate phase for the left extension of $\Psi_{2:4}$ by $\tau_1 = (120, \phi_1)$, with the phases of $\Psi_{2:4}$ fixed as in Fig. 2. Since $(-60) \bmod 60 = 60 \bmod 60 = 0$, both first-read times yield the singleton candidate phase set $\mathcal{C} = \{0\}$. With $\phi_1 = 0$, the latest connectable jobs of τ_1 write at the first-read times $s^{(0)} - H$ and $s^{(1)} - H$ of the shown H-shifted copies of $\psi^{(0)}$ and $\psi^{(1)}$, respectively, yielding zero wait time.

Θ is attained. Accordingly, we seek an optimal phase vector $\vec{\varphi}^\star \in \Theta$ satisfying

$$\mathrm{Lat}(\Psi_{1:N}; \vec{\varphi}^\star) = \min_{\vec{\varphi}_{1:N} \in \Theta} \mathrm{Lat}(\Psi_{1:N}; \vec{\varphi}_{1:N}), \tag{2}$$

together with the corresponding minimum E2E latency $\mathrm{Lat}(\Psi_{1:N}; \vec{\varphi}^\star)$.

IV. Candidate Phases via Zero-Wait Alignment

A. Wait Times and Candidate Phases

Throughout Section IV, consider a task subchain $\Psi_{a:b}$, where $1 \le a \le b \le N$, containing $\tau_{\widehat{p}}$, with a fixed partial phase vector $\vec{\varphi}_{a:b}$. Assume that either $b < N$ or $a > 1$. We consider extending this task subchain by appending τ_{b+1} when $b < N$ (a right extension), or by prepending τ_{a-1} when $a > 1$ (a left extension).

We use the quantities defined in Section III and omit the subscript $a:b$ from H, M, $\psi^{(j)}$, $s^{(j)}$, $w^{(j)}$, $\mathcal{S}$, and $\mathcal{W}$ when no confusion arises. For the chosen extension direction, let $e^{(j)}$ denote the endpoint of $\psi^{(j)}$ adjacent to the added task and $\bar{e}^{(j)}$ the opposite endpoint, i.e., $(e^{(j)}, \bar{e}^{(j)}) = (w^{(j)}, s^{(j)})$ for a right extension and $(s^{(j)}, w^{(j)})$ for a left extension. Let $\mathcal{E} = (e^{(0)}, \ldots, e^{(M-1)})$. In expressions with $\pm$ or $\mp$, the upper sign applies to a right extension and the lower sign to a left

extension. Let T and ϕ denote the period and phase of the added task, respectively, and let $g = \gcd(H, T)$. Let U denote the period of the boundary task of $\Psi_{a:b}$ adjacent to the added task ($U = T_b$ for a right extension and $U = T_a$ for a left extension), and let $\gamma = \gcd(U, T)$. Here, $x \bmod u$ denotes the remainder in $[0, u)$, also for negative x.

Each copy of $\psi^{(j)}$ is extended to the earliest connectable job of the successor τ_{b+1} for a right extension, or backward to the latest connectable job of the predecessor τ_{a-1} for a left extension. We define the *wait time* of a copy of $\psi^{(j)}$ as the duration from the copy's last write-event to the read-event of the selected successor job for a right extension, and as the duration from the write-event of the selected predecessor job to the copy's first read-event for a left extension. For the $m \cdot H$-shifted copy of $\psi^{(j)}$, where $m \in \mathbb{Z}$, let $\delta_m^{(j)}(\phi)$ denote its wait time. Since the added task's read- and write-events occur at ϕ plus integer multiples of T,

$$\delta_m^{(j)}(\phi) = \big(\pm(\phi - e^{(j)} - m \cdot H)\big) \bmod T.$$

Let $\delta^{(j)}(\phi) = \min_{m \in \mathbb{Z}} \delta_m^{(j)}(\phi)$. Let $L = \mathrm{Lat}(\Psi_{a:b}; \vec{\varphi}_{a:b}) = \max_{0 \le j < M}\{w^{(j)} - s^{(j)}\}$, and denote the E2E latency after assigning phase ϕ to the added task by

$$L^+(\phi) = \begin{cases} \mathrm{Lat}(\Psi_{a:b+1}; (\vec{\varphi}_{a:b}, \phi)), & \text{right extension,} \\ \mathrm{Lat}(\Psi_{a-1:b}; (\phi, \vec{\varphi}_{a:b})), & \text{left extension.} \end{cases}$$

We narrow the candidate space of ϕ (ϕ_{b+1} when appending τ_{b+1}, or ϕ_{a-1} when prepending τ_{a-1}) in two steps: from $\mathbb{R}$ to $[0, g)$ (Lemma 1), and then to the finite set (Theorem 1),

$$\mathcal{C} = \{e \bmod g \mid e \in \mathcal{E}\}. \tag{3}$$

We call $\mathcal{C}$ the set of *candidate phases*. By Lemma 2, $1 \le |\mathcal{C}| \le \min(g/\gamma, M)$.

B. Characterization of Task-Subchain Extensions

Lemma 1 below extends a known phase equivalence. Martinez et al. [16] observed that, once the phases of $\tau_1, \ldots, \tau_{i-1}$ are set, it suffices to consider phases of τ_i in $[0, \gcd(T_i, \mathrm{lcm}(T_1, \ldots, T_{i-1})))$. Wang et al. [15, Lemma 2] proved the corresponding E2E-latency invariance for integer phases in the zeroLET model. Here, we establish this equivalence for real-valued phases and for a task subchain extended in either direction, supporting task-extension orders that start from $\tau_{\widehat{p}}$.

Lemma 1 (Phase equivalence): With $\vec{\varphi}_{a:b}$ fixed, every extension $\vec{\varphi}_{1:N}$ assigning phase ϕ to the added task has an E2E-latency-equivalent extension $\vec{\varphi}'_{1:N}$ with phase $\phi' = \phi \bmod g$ assigned to the added task.

Proof: For any such $\vec{\varphi}_{1:N}$, Bézout's identity gives $\phi - \phi' = k \cdot g = v \cdot H + u \cdot T$ for some $u, v, k \in \mathbb{Z}$. Subtract $v \cdot H$ from every phase outside $\Psi_{a:b}$. By the H-periodicity of $\Psi_{a:b}$, the resulting task-wise sets of event times are the same as those obtained by subtracting $v \cdot H$ from every phase of $\Psi_{1:N}$, after the corresponding jobs are reindexed. Since the E2E latency depends only on these event-time sets, it is unchanged. The phase of the added task is $\phi - v \cdot H = \phi' + u \cdot T$; replacing it by ϕ' only reindexes that task's jobs, yielding $\vec{\varphi}'_{1:N}$. ∎

Lemma 2 (Candidate phase cardinality): All endpoints in $\mathcal{E}$ and candidate phases in $\mathcal{C}$ are congruent modulo γ, and $1 \le |\mathcal{C}| \le \min(g/\gamma, M)$.

Proof: All endpoints are congruent modulo U as events of the same boundary task. Since $U \mid H$, $\gcd(U, g) = \gamma$, so both the endpoints and their residues modulo g are congruent modulo γ. There are g/γ such values in $[0, g)$; combining this with $1 \le |\mathcal{C}| \le M$ proves the bound. ∎

Lemma 3 (Wait times): For each $\psi^{(j)}$, the wait times of its H-shifted copies satisfy

$$\{\delta_m^{(j)}(\phi) \mid m \in \mathbb{Z}\} = \{\delta^{(j)}(\phi) + q \cdot g \mid q \in \mathbb{Z},\ 0 \le q < T/g\},$$

where the minimum wait time is

$$\delta^{(j)}(\phi) = \big(\pm(\phi - e^{(j)})\big) \bmod g. \tag{4}$$

Proof: Fix j and ϕ. Since $g = \gcd(H, T)$, the distinct residues of either $m \cdot H$ or $-m \cdot H$ modulo T, for $m \in \mathbb{Z}$, are $0, g, \ldots, T - g$. The wait times are obtained by adding $\pm(\phi - e^{(j)})$ to these residues and reducing modulo T. This preserves distinctness, yielding exactly T/g distinct wait times in $[0, T)$. All are congruent modulo g to $\pm(\phi - e^{(j)})$. There are exactly T/g values in $[0, T)$ satisfying the corresponding congruence: its least nonnegative residue plus $q \cdot g$, for $q = 0, \ldots, T/g - 1$. Hence all these values occur. ∎

Corollary 1 (Exact latency after extension): For each $\psi^{(j)}$, $\max_{m \in \mathbb{Z}} \delta_m^{(j)}(\phi) = \delta^{(j)}(\phi) + T - g$, and

$$L^+(\phi) = 2 \cdot T - g + \max_{0 \le j < M} \big\{w^{(j)} - s^{(j)} + \delta^{(j)}(\phi)\big\}. \tag{5}$$

Proof: For $\psi^{(j)}$, Lemma 3 gives the stated maximum wait among its copies in the extended hyperperiod. The extension adds this wait and one LET interval length T to $w^{(j)} - s^{(j)}$. Maximizing over j proves Eq. (5). ∎

Corollary 2 (Latency bounds after extension): For every phase ϕ of the added task, $L^+(\phi) \ge L + 2 \cdot T - g$. If $\phi \in \mathcal{C}$, then $L^+(\phi) \le L + 2 \cdot T - \gamma$.

Proof: The lower bound follows from Eq. (5) and $\delta^{(j)}(\phi) \ge 0$. For $\phi \in \mathcal{C}$, Lemma 2 and Eq. (4) imply $\delta^{(j)}(\phi) \in \gamma\mathbb{Z} \cap [0, g)$ for every j; since $\gamma \mid g$, $\delta^{(j)}(\phi) \le g - \gamma$. Using this bound and $w^{(j)} - s^{(j)} \le L$ in Eq. (5) proves the upper bound. ∎

For $\phi \in [0, g)$, Eqs. (3) and (4) give $\phi \in \mathcal{C}$ if and only if $\delta^{(j)}(\phi) = 0$ for some j, equivalently, if and only if at least one copy of some $\psi^{(j)}$ has zero wait time.

For example, appending τ_5 with $T_5 = 120$ to $\Psi_{2:4}$ in Fig. 2 gives $H = 300$, $g = 60$, and $\mathcal{E} = \mathcal{W} = (400, 600)$, hence $\mathcal{C} = \{40, 0\}$. In Fig. 3, $\phi_5 = 40$ and $\phi_5 = 0$ give zero wait at $w^{(0)}$ and $w^{(1)}$, respectively. For the left extension by τ_1 with $T_1 = 120$, $g = \gamma = 60$ and $\mathcal{E} = \mathcal{S} = (-60, 60)$ yield the singleton $\mathcal{C} = \{0\}$ (Lemma 2). The phase $\phi_1 = 0$ gives zero wait at $s^{(0)} - H$ and $s^{(1)} - H$ for the copies shown in Fig. 4.

C. Global Optimality and E2E Latency Bounds

Theorem 1 (Zero-wait alignment): Every complete phase vector extending $\vec{\varphi}_{a:b}$ has a counterpart with no greater E2E latency that retains $\vec{\varphi}_{a:b}$ and assigns $\phi \in \mathcal{C}$ to the added task. Thus, restricting ϕ to $\mathcal{C}$ preserves global optimality for either extension direction.

Proof: By Lemma 1, assume $\phi \in [0, g)$. If $\phi \in \mathcal{C}$, the claim holds. Otherwise, let $\Delta = \min_{0 \le j < M} \delta^{(j)}(\phi) > 0$. Add $\mp\Delta$ to the phases of all tasks on the added side of $\Psi_{a:b}$ ($\tau_{b+1}, \ldots, \tau_N$ or $\tau_1, \ldots, \tau_{a-1}$). By the choice of Δ, the selected job of the added task remains unchanged for every copy of each $\psi^{(j)}$ as the wait times decrease by Δ, with at least one becoming zero. All connections among the shifted tasks are preserved. Consequently, every complete $\tau_{\widehat{p}}$-partitioned job chain retains its endpoint on the opposite side, while its endpoint on the added side changes by $\mp\Delta$, decreasing its length by Δ and hence the E2E latency by Δ. By Eq. (4), $(\phi \mp \Delta) \bmod g \in \mathcal{C}$; applying Lemma 1 replaces $\phi \mp \Delta$ by this residue without changing the E2E latency. ∎

Corollary 3 (Existence of a global optimum): For any task-extension order that starts from $\tau_{\widehat{p}}$ with $\phi_{\widehat{p}} = 0$ and adds each subsequent task by a left or right extension, at least one complete phase vector generated by recursively selecting the phase of each added task from the corresponding set $\mathcal{C}$ in Eq. (3) is globally optimal over Θ.

Proof: The complete phase vectors generated by all such selections form a finite nonempty collection because every set $\mathcal{C}$ is finite and nonempty. By recursive application of Theorem 1, every vector in Θ has a counterpart in this collection with no greater E2E latency. Therefore, a minimum-latency vector in this collection is globally optimal over Θ. ∎

For any task-extension order $(\sigma_1, \ldots, \sigma_N)$ with $\sigma_1 = \widehat{p}$ that uses left or right extensions, let $H_d = \mathrm{lcm}(T_{\sigma_1}, \ldots, T_{\sigma_d})$ for $1 \le d \le N$. For $2 \le d \le N$, let $U_d = T_{\sigma_d \mp 1}$ denote the period of the adjacent boundary task, and let $g_d = \gcd(H_{d-1}, T_{\sigma_d})$ and $\gamma_d = \gcd(U_d, T_{\sigma_d})$.

Theorem 2 (Bounds on the minimum E2E latency): For any such task-extension order,

$$\begin{aligned} 2 \cdot \sum_{i=1}^{N} T_i - \sum_{d=2}^{N} g_d &\le \min_{\vec{\varphi} \in \Theta} \mathrm{Lat}(\Psi_{1:N}; \vec{\varphi}) \\ &\le 2 \cdot \sum_{i=1}^{N} T_i - \sum_{i=1}^{N-1} \gcd(T_i, T_{i+1}). \end{aligned} \tag{6}$$

Proof: By Corollary 3, at least one complete phase vector generated along the given task-extension order is globally optimal over Θ. Starting from its initial latency $2{\cdot}T_{\sigma_1}$, applying the bounds in Corollary 2 at every extension gives Eq. (6), since $(\sigma_1, \ldots, \sigma_N)$ is a permutation of $(1, \ldots, N)$ and each adjacent task pair is connected by exactly one extension. ∎

The maximum of the lower bounds in Eq. (6) over all such task-extension orders is itself a valid lower bound. For max-harmonic CECs, described in Section VI-B, we have $H_d = T_{\widehat{p}}$ and $g_d = T_{\sigma_d}$ for $d \ge 2$. Hence, the lower bound in Eq. (6) reduces to $\sum_{i=1}^{N} T_i + T_{\widehat{p}}$, matching the minimum E2E latency established in [7, Lemma 12 and Theorem 13].

Corollary 4 (Optimality of the zero phase vector): For any such task-extension order, if $g_d \mid U_d$ for every $d = 2, \ldots, N$, the zero phase vector $\vec{\varphi}^\star = (0, \ldots, 0)$ is globally optimal, and its E2E latency is

$$\mathrm{Lat}(\Psi_{1:N}; \vec{\varphi}^\star) = 2 \cdot \sum_{i=1}^{N} T_i - \sum_{i=1}^{N-1} \gcd(T_i, T_{i+1}). \tag{7}$$

Proof: Since $U_d \mid H_{d-1}$, $\gamma_d = \gcd(U_d, g_d) \mid g_d$. Thus, the bounds in Eq. (6) coincide if and only if $\sum_{d=2}^{N}(g_d - \gamma_d) = 0$, equivalently, $g_d \mid U_d$ at every extension. Under this condition, Lemma 2 gives a singleton candidate set. If all assigned phases are zero, every relevant endpoint is a multiple of U_d, so Eq. (3) gives $\mathcal{C} = \{0\}$. Induction from $\phi_{\widehat{p}} = 0$ yields the zero phase vector as the unique generated complete phase vector. Corollary 3 establishes its optimality, and the coincident bounds give Eq. (7). ∎

For every CEC with $N = 2$, $U_2 = T_{\widehat{p}}$ and $g_2 = \gcd(T_1, T_2) \mid U_2$. Hence $\vec{\varphi}^\star = (0, 0)$ is globally optimal, with minimum E2E latency $2{\cdot}T_1 + 2{\cdot}T_2 - \gcd(T_1, T_2)$. The camera-based control CEC in Fig. 1 also satisfies the divisibility condition of Corollary 4, yielding a globally optimal zero phase vector with E2E latency 5040 ms, as illustrated in Fig. 1(a). Pairwise coprime task periods also satisfy this condition, since $g_d = 1$ at every extension and $\gcd(T_i, T_{i+1}) = 1$ for every adjacent task pair; the zero phase vector is therefore globally optimal with E2E latency $2 \cdot \sum_{i=1}^{N} T_i - (N-1)$.

V. Search Algorithm and Correctness

A. Exact Branch-and-Bound Search

Since the zero phase vector is globally optimal for every two-task CEC by Corollary 4, we formulate the search algorithm for $N \ge 3$. As a deterministic convention, we adopt the right-then-left task-extension order $(\sigma_1, \ldots, \sigma_N) = (\widehat{p}, \widehat{p}+1, \ldots, N, \widehat{p}-1, \ldots, 1)$. This task-extension order does not change the task order $\tau_1 \to \cdots \to \tau_N$. For this extension order, we define a finite *candidate-phase tree*: the root at depth one represents $\phi_{\widehat{p}} = 0$, and each node at depth d represents a phase assignment to $\tau_{\sigma_1}, \ldots, \tau_{\sigma_d}$. For $d < N$, each node has one outgoing edge per phase in $\mathcal{C}$ of Eq. (3); each edge assigns that phase to $\tau_{\sigma_{d+1}}$ to form a child. Nodes at depth N are leaves and represent complete phase vectors, at least one of which is globally optimal by Corollary 3. We explore this tree by depth-first branch-and-bound.

At depth d, let $a_d = \min\{\sigma_1, \ldots, \sigma_d\}$ and $b_d = \max\{\sigma_1, \ldots, \sigma_d\}$; the assigned task subchain is $\Psi_{a_d:b_d}$. For the adopted right-then-left order, equivalently, $a_d = \min(\widehat{p}, N - d + 1)$ and $b_d = \min(\widehat{p} + d - 1, N)$. Then, $H_d = H_{a_d:b_d}$. Let $M_d = M_{a_d:b_d}$, $\mathcal{Q}_d = (\mathcal{S}_{a_d:b_d}, \mathcal{W}_{a_d:b_d})$, and $L_d = \mathrm{Lat}(\Psi_{a_d:b_d}; \vec{\varphi}_{a_d:b_d})$. At a fixed depth, we omit the subscript $a_d : b_d$ from $\psi^{(j)}$, $s^{(j)}$, $w^{(j)}$, $\mathcal{S}$, and $\mathcal{W}$. Thus, $\mathcal{Q}_d = (\mathcal{S}, \mathcal{W})$ stores the endpoint pairs of the M_d job chains, and $L_d = \max_{0 \le j < M_d}(w^{(j)} - s^{(j)})$.

For the extension from depth d to depth $d+1$, where $1 \le d < N$, the extension is right if $\sigma_{d+1} = b_d + 1$ and left if $\sigma_{d+1} = a_d - 1$. Using the same sign convention as in Section IV, the adjacent boundary task is $\tau_{\sigma_{d+1} \mp 1}$. Accordingly, $U_{d+1} = T_{\sigma_{d+1} \mp 1}$, $g_{d+1} = \gcd(H_d, T_{\sigma_{d+1}})$, and $\gamma_{d+1} = \gcd(U_{d+1}, T_{\sigma_{d+1}})$.

The search algorithm combines two independently applicable accelerations: *lower-bound pruning* (LBP) and *singleton acceleration* (SA). During the search, let $L^\star$ denote the smallest E2E latency among the complete phase vectors found so far, called the *incumbent latency*; before any complete phase vector is found, $L^\star = +\infty$.

The algorithm in Fig. 5 first precomputes the period-dependent quantities and SA flags, initializes the root state and incumbent, and starts the search (lines 1–7). At depth d, if $g_k \mid U_k$ for every remaining extension, it completes the unique path and records its exact E2E latency; we call this procedure *path-level SA* (lines 10–12; Corollary 5). Otherwise, if $g_{d+1} \mid U_{d+1}$, it obtains the unique candidate phase and the exact latency L' of the extended task subchain; we call this procedure *step-level SA* (lines 14–16; Lemma 6). If this condition fails, it forms $\mathcal{C}$ by scanning $\mathcal{E}$ and skipping duplicate residues, as specified by Eq. (3) (line 17).

Candidates are examined in first-occurrence order (line 18). For each candidate, the algorithm assigns the task phase (line 19) and, when step-level SA is inapplicable, computes L' (line 20; Corollary 1). If L' plus the lower bound for the remaining extensions is no smaller than $L^\star$, the candidate is discarded before constructing its endpoint state; we call this test *child-level LBP* (line 21; Lemma 4). For each surviving candidate, the algorithm updates $L^\star$ and records the phase vector if it is complete; otherwise, it constructs the child state and recurses (lines 22–23). It stops examining the remaining candidates when $L^\star$ reaches the state's E2E-latency lower bound; we call this test *state-level LBP* (line 24).

Lemma 4 (Admissible pruning bound): Let

$$B_d = \sum_{k=d+1}^{N} (2 \cdot T_{\sigma_k} - g_k), \qquad B_N = 0. \tag{8}$$

Every complete phase vector reachable from endpoint state $\mathcal{Q}_d$ has E2E latency at least $L_d + B_d$.

Proof: By the lower bound in Corollary 2, extension k increases the E2E latency by at least $2 \cdot T_{\sigma_k} - g_k$. Summing over $k = d+1, \ldots, N$ proves the claim. ∎

Lemma 5 (Exact endpoint-state extension): Given endpoint state $\mathcal{Q}_d = (\mathcal{S}, \mathcal{W})$ and a phase ϕ for the added task $\tau_{\sigma_{d+1}}$ of period $T = T_{\sigma_{d+1}}$, let $\delta_q^{(j)}(\phi)$ denote the wait time of the $q \cdot H_d$-shifted copy of $\psi^{(j)}$, with $H = H_d$. For integers j and q with $0 \le j < M_d$ and $0 \le q < H_{d+1}/H_d$, let $k = j + q \cdot M_d$. Then EXTEND constructs the exact child endpoint state $\mathcal{Q}_{d+1}$ by

$$\begin{aligned} e'^{(k)} &= e^{(j)} + q \cdot H_d \pm \big(\delta_q^{(j)}(\phi) + T\big), \\ \bar{e}'^{(k)} &= \bar{e}^{(j)} + q \cdot H_d, \end{aligned} \tag{9}$$

where $e'^{(k)}$ and $\bar{e}'^{(k)}$ denote the endpoints of the extended job chain on the same sides as $e^{(j)}$ and $\bar{e}^{(j)}$.

Proof: Each job chain $\psi^{(k)}_{a_{d+1}:b_{d+1}}$ is obtained by extending the $q \cdot H_d$-shifted copy of $\psi^{(j)}_{a_d:b_d}$, where $k = j + q \cdot M_d$. The endpoint of that copy adjacent to the added task moves outward by the wait time $\delta_q^{(j)}(\phi)$ and one LET interval length T, while its opposite endpoint is unchanged. Since $M_{d+1} = (H_{d+1}/H_d) \cdot M_d$, the pairs (j, q) enumerate exactly once all $\psi^{(k)}_{a_{d+1}:b_{d+1}}$ with $0 \le k < M_{d+1}$. ∎

Lemma 6 (Singleton extension): For the extension assigning τ_{σ_k}, if $g_k \mid U_k$, then, for every endpoint state $\mathcal{Q}_{k-1}$ reachable at depth $k-1$, the candidate phase set in Eq. (3) is a singleton:

$$\mathcal{C} = \{\phi_{\sigma_k \mp 1} \bmod g_k\}.$$

Require: Periods $T_1, \ldots, T_N \in \mathbb{N}_{>0}$ and $N \ge 3$
Ensure: An optimal phase vector $\vec{\varphi}^\star$ and latency $L^\star$
1: Determine $\widehat{p}$ and the task-extension order $\sigma_1, \ldots, \sigma_N$
2: Precompute H_d, M_d, B_d $(1 \le d \le N)$, and g_d, U_d $(2 \le d \le N)$ // Lem. 4
3: $\mathrm{sgl}_k \leftarrow [g_k \mid U_k]$ $(2 \le k \le N)$ // Lem. 6
4: $\mathrm{path}_d \leftarrow \bigwedge_{k=d+1}^{N} \mathrm{sgl}_k$ $(1 \le d < N)$ // Cor. 5
5: $\mathcal{Q}_1 \leftarrow ((0), (2 \cdot T_{\widehat{p}}))$
6: $\vec{\varphi} = (\phi_1, \ldots, \phi_N) \leftarrow (0, \ldots, 0)$; $\vec{\varphi}^\star \leftarrow (0, \ldots, 0)$; $L^\star \leftarrow +\infty$
7: SEARCH$(1, \mathcal{Q}_1, 2 \cdot T_{\widehat{p}})$
8: **return** $(\vec{\varphi}^\star, L^\star)$
9: **procedure** SEARCH$(d, \mathcal{Q}_d, L_d)$
10: **if** path_d **then** // Path-level SA; Cor. 5
11: **for** $k = d+1, \ldots, N$ **do** $\phi_{\sigma_k} \leftarrow \phi_{\sigma_k \mp 1} \bmod g_k$ // Lem. 6
12: $L^\star \leftarrow L_d + B_d$; $\vec{\varphi}^\star \leftarrow \vec{\varphi}$; **return**
13: $\mathcal{E} \leftarrow \mathcal{W}$ if right, otherwise $\mathcal{S}$, where $(\mathcal{S}, \mathcal{W}) = \mathcal{Q}_d$
14: **if** sgl_{d+1} **then** // Step-level SA; Lem. 6
15: $\mathcal{C} \leftarrow \{\phi_{\sigma_{d+1} \mp 1} \bmod g_{d+1}\}$
16: $L' \leftarrow L_d + 2 \cdot T_{\sigma_{d+1}} - g_{d+1}$ // Eq. (10)
17: **else** $\mathcal{C} \leftarrow \{e \bmod g_{d+1} \mid e \in \mathcal{E}\}$ // Eq. (3)
18: **for** each $\phi \in \mathcal{C}$ in first-occurrence order **do**
19: $\phi_{\sigma_{d+1}} \leftarrow \phi$
20: **if** $\neg\mathrm{sgl}_{d+1}$ **then** $L' \leftarrow L^+(\phi_{\sigma_{d+1}})$ // Eq. (5), Cor. 1
21: **if** $L' + B_{d+1} \ge L^\star$ **then continue** // Child-level LBP; Lem. 4
22: **if** $d + 1 = N$ **then** $\{L^\star \leftarrow L';\ \vec{\varphi}^\star \leftarrow \vec{\varphi}\}$
23: **else** $\{\mathcal{Q}' \leftarrow$ EXTEND$(d, \mathcal{Q}_d, \phi_{\sigma_{d+1}})$; SEARCH$(d+1, \mathcal{Q}', L')\}$ // Lem. 5
24: **if** $L_d + B_d = L^\star$ **then break** // State-level LBP; Lem. 4
25: **return**
26: **end procedure**

Fig. 5. Exact depth-first search with singleton acceleration (SA) and lower-bound pruning (LBP). For $2 \le k \le N$, the extension assigning τ_{σ_k} is right for $k \le N - \widehat{p} + 1$ and left otherwise. Signs follow the convention in Section IV. The Boolean flags sgl_k and path_d record whether the divisibility condition holds at extension k for step-level SA and at every remaining extension for path-level SA, respectively. Square brackets denote Boolean truth values.

Assigning this unique phase yields the E2E latency

$$L_k = L_{k-1} + 2 \cdot T_{\sigma_k} - g_k. \tag{10}$$

Proof: The divisibility condition $g_k \mid U_k$ is equivalent to $g_k/\gamma_k = 1$, so Lemma 2 shows that the set $\mathcal{C}$ in Eq. (3) is a singleton. All boundary endpoints are congruent to $\phi_{\sigma_k \mp 1}$ modulo U_k, hence modulo g_k, which identifies the unique phase as $\phi_{\sigma_k \mp 1} \bmod g_k$. Since $\gamma_k = g_k$, the bounds in Corollary 2 coincide, yielding Eq. (10). ∎

We call an extension satisfying $g_k \mid U_k$ a *singleton extension* (Lemma 6). This divisibility condition depends only on the periods and can therefore be precomputed. For $N = 4$, the final left extension in Fig. 4 has $g_4 = U_4 = 60$. The boundary task has phase 0, so $\mathcal{C} = \{0\}$, and Eq. (10) gives $L_4 = 540 + 2 \cdot 120 - 60 = 720$.

Corollary 5 (Unique phase completion): If $g_k \mid U_k$ for every $k = d+1, \ldots, N$, the subtree rooted at any depth-d node consists of a single path. The unique completion along this path has the exact latency

$$L_N = L_d + B_d.$$

Proof: Lemma 6 gives one candidate phase and exact growth $2 \cdot T_{\sigma_k} - g_k$ at every remaining extension. Hence, the remaining extensions form a single path, and summing these increments yields $L_N = L_d + B_d$. ∎

Path-level SA therefore assigns each remaining phase from the phase of its boundary task using Lemma 6 (line 11), without constructing endpoint states, and records the exact completion latency $L_d + B_d$ (line 12).

Theorem 3 (Global optimality): For $N \geq 3$, the proposed algorithm in Fig. 5 returns a globally optimal phase vector $\vec{\varphi}^{\star} \in \Theta$ and its E2E latency.

Proof: By Corollary 3, the finite candidate-phase tree contains a global optimum. The root state $\mathcal{Q}_1 = ((0), (2 \cdot T_{\widehat{p}}))$ is exact by the partitioned-job-chain definition with $\phi_{\widehat{p}} = 0$. Lemma 5 preserves exactness when constructing the child state in line 23. Lemma 6 justifies step-level SA (lines 14–16), while Corollary 1 gives the other child latencies.

At entry to each SEARCH call,

$$\mathcal{I}(d): \qquad L_d + B_d < L^{\star}.$$

This holds initially and at every recursive call by the child-level LBP test. The working phase vector retains the current assignments at depth d. In increasing order of k, the boundary task $\tau_{\sigma_k \mp 1}$ already has an assigned phase. By Lemma 6 and Corollary 5, path-level SA (lines 10–12) recovers the unique completion with latency $L_d + B_d$; $\mathcal{I}(d)$ therefore guarantees a strict improvement.

During a call, descendant updates may reduce $L^{\star}$ to $L_d + B_d$, but not below it (Lemma 4). Thus state-level LBP is safe; the same lemma applied to each child justifies child-level LBP. Every finite incumbent is attained by a complete phase vector (lines 12 and 22), so both tests are safe at equality. The finite search therefore terminates without discarding any completion that could improve the incumbent, yielding a global optimum. Only nonroot phases are reassigned after line 6, so the returned vector belongs to Θ. ∎

Since the periods and root endpoints are integers, Eq. (9) and Eq. (3) preserve integer endpoints and candidate phases at every depth. Thus, although Θ permits real phases, the candidate-phase tree retains an integer-valued optimum.

B. Complexity Analysis

On the integer phase grid, Lemma 1 restricts each added task's phase to the g_d integers in $[0, g_d)$. With the root phase fixed at zero, an exact search using only Lemma 1 considers

$$\prod_{d=2}^{N} g_d = \frac{\prod_{i=1}^{N} T_i}{H_{1:N}} \tag{11}$$

phase vectors. The equality follows by telescoping $H_d \cdot g_d = H_{d-1} \cdot T_{\sigma_d}$, as in [15], [16]. The right-hand side counts the classes, each containing $H_{1:N}$ vectors related by a common integer time shift, among the $\prod_{i=1}^{N} T_i$ integer phase vectors with each ϕ_i taken modulo T_i. This is the number of non-equivalent integer phase vectors enumerated by Wang et al. [15] in the zeroLET model and by the full-depth search of Martinez et al. [16].

By Lemma 2, each extension from depth $d-1$ to depth d has at most $\min(g_d/\gamma_d, M_{d-1})$ candidate phases. Thus, the candidate-phase tree has at most $\prod_{k=2}^{d} \min(g_k/\gamma_k, M_{k-1})$ nodes at depth d and at most

$$\prod_{d=2}^{N} \min(g_d/\gamma_d, M_{d-1}) \tag{12}$$

leaves. This bound can vary with the task-extension order through g_d/γ_d and M_{d-1}. Each factor is at most g_d, so Eq. (12) never exceeds Eq. (11) and is strictly smaller whenever $\gamma_d > 1$ or $M_{d-1} < g_d$ for some d. Refining the integer time unit by an integer factor $c > 1$ scales both g_d and γ_d by c, leaving g_d/γ_d and M_d unchanged. Hence, the bound in Eq. (12) is unchanged, whereas Eq. (11) increases by a factor of c^{N-1}. The corresponding sets $\mathcal{C}$ in Eq. (3) retain their cardinalities.

Assuming constant-time integer arithmetic and expected constant-time hashing, candidate generation at depth $d-1$ costs $O(M_{d-1})$. Each candidate costs $O(M_{d-1})$ for exact latency evaluation by Corollary 1 and $O(M_d)$ if its child endpoint state is constructed by Lemma 5. Charging generation to parent nodes and the other costs to child nodes, with $M_{d-1} \leq M_d$, gives the worst-case expected running time

$$O\left(\sum_{d=1}^{N} M_d \prod_{k=2}^{d} \min(g_k/\gamma_k, M_{k-1}) + N \cdot \log(1 + T_{\widehat{p}})\right),$$

where the product is one for $d = 1$. The $N \cdot \log(1 + T_{\widehat{p}})$ term accounts for preprocessing gcd and lcm computations. The assignments $\vec{\varphi}^{\star} \leftarrow \vec{\varphi}$ in Fig. 5 (lines 12 and 22) are implemented by recording phases during backtracking rather than copying the vector. Path-level SA's $O(N-d)$ phase recovery is charged to its unique path in the full candidate-phase tree.

For space, endpoint-state and candidate-phase buffers use $O(M_d)$ per depth $d < N$, totaling $O(\sum_{d=1}^{N-1} M_d)$; no additional endpoint states are constructed at leaves or during path-level SA. These worst-case bounds remain valid with SA and LBP, which reduce search work, as evaluated in Section VI-D.

VI. EVALUATION

A. Experimental Setup

The algorithm implementations, experimental setup details, and data described in this section are available in our GitHub repository.[1] The compared algorithms are as follows.

- *Günzel'25* [7] computes an optimal phase vector in closed form for max-harmonic and $(2, k)$-max-harmonic period sets. We include this method only for CECs satisfying these period conditions.
- *Wang'26* [15] searches all non-equivalent integer phase vectors in the zeroLET model, with the first task's phase fixed at zero. For each phase vector, it computes the E2E latency from the immediate forward job chains within one hyperperiod. From an optimal phase vector in the zeroLET model, it recovers an optimal phase vector and the minimum E2E latency for the original LET-based CEC. Because [15] evaluates E2E latency on an integer time grid, its value is one time unit smaller than that defined by Eq. (1); we add this constant when comparing minimum E2E latencies.
- *Proposed* is the complete algorithm of Section V, with singleton acceleration (SA) and lower-bound pruning (LBP).
- *Proposed without SA* retains LBP and disables SA.

[1] https://github.com/SNU-RTOS/accelerated-phasing (release `accelerated-phasing-v1.1`).

TABLE III
EVALUATED PHASE VECTOR COUNTS AND EXECUTION TIMES AT $N = 5$ (MEDIAN AND P99 PER WORKLOAD). TIMES ARE IN μS.

	Semi-harmonic				(x, y)-hyperperiod				Randomly sampled			
	Counts		Time		Counts		Time		Counts		Time	
Method	Median	P99	Median	P99	Median	P99	Median	P99	Median	P99	Median	P99
Wang'26	$2 \cdot 10^4$	$5 \cdot 10^8$	$2.02 \cdot 10^3$	$4.01 \cdot 10^7$	$1.44 \cdot 10^5$	$5.49 \cdot 10^7$	$6.57 \cdot 10^4$	$2.38 \cdot 10^7$	24	4,116	206	$3.88 \cdot 10^4$
Proposed	1	1	0.347	0.548	3	405	0.918	24.0	2	42	1.09	152
Speedup	—	—	$5.81 \cdot 10^3$	$7.32 \cdot 10^7$	—	—	$7.16 \cdot 10^4$	$9.90 \cdot 10^5$	—	—	190	255

Wang'26 evaluated phase vector counts are obtained analytically. Median and P99 speedups are the ratios of the corresponding Wang'26 and Proposed execution-time statistics within each workload.

- *Proposed without LBP* retains SA and disables LBP.
- *Proposed without SA and LBP* disables both accelerations. It explores the full candidate-phase tree underlying the complexity bound in Section V-B, providing the unaccelerated execution-time reference in Fig. 7.

We implement Wang'26 [15] and Günzel'25 [7] in C++ based on the authors' publicly available source code.[2]

Experiments run on Ubuntu 22.04.5 with Linux 6.8.0-138-generic, an Intel Core i7-14650HX processor, and 31 GiB RAM. All methods are single-threaded C++17 programs compiled with g++ 11.4.0 using `-std=c++17 -O3 -DNDEBUG -g0 -s`. The methods run sequentially on the same isolated physical P-core.

We evaluate one application-derived workload comprising five CECs. We also evaluate three synthetic workloads: (i) semi-harmonic, (ii) (x, y)-hyperperiod, and (iii) randomly sampled. For each synthetic workload and task-chain length, we generate 500 distinct CECs. Section VI-B describes each workload in detail.

To assess how efficiently the methods compute an optimal phase vector and its minimum E2E latency, we compare two metrics: the evaluated phase vector count and algorithm execution time. First, for each CEC, the *evaluated phase vector count* is the number of complete phase vectors $\vec{\varphi}_{1:N}$ for which an algorithm computes the latency during optimization. Wang'26 evaluates all non-equivalent integer phase vectors in the zeroLET model, whose count is given by Eq. (11). Günzel'25 evaluates one optimal phase vector per applicable CEC. Proposed without SA and LBP evaluates every leaf of the candidate-phase tree, with the count bounded by Eq. (12). LBP can reduce the evaluated phase vector count by pruning before the exact latency is computed; complete phase vectors rejected after this computation are still counted. Partial phase vectors are excluded from this count. SA preserves the evaluated phase vector count while reducing intermediate work. This count is independent of hardware performance, but does not represent total computational work.

Second, we measure each method's execution time for each applicable CEC using `std::chrono::steady_clock`. The timed interval includes fresh solver construction, algorithm execution, and result extraction, excluding process startup and I/O. If the initial execution time is below 1 s, we perform additional runs, with more repetitions for shorter initial execution times. For each method and CEC, we report the median execution time across all runs, including the initial run. For each synthetic workload and task-chain length, we omit all Wang'26 execution-time measurements if the estimated total execution time for running each of the 500 CECs once exceeds one day.

The accompanying GitHub repository provides all per-CEC results used in this evaluation. It also documents validation procedures and results for our Wang'26 and Günzel'25 implementations, workload generation, the detailed evaluation protocol, and the execution-time prediction model used to determine which Wang'26 measurements to omit.

[2] [15]: https://github.com/wsm6636/ZEROLET;
[7]: https://github.com/ESRTS/OptimalTaskPhasing.

B. Workloads

For the three synthetic workloads, we evaluate task-chain lengths $N = 3, \ldots, 8$ with integer task periods. Industrial benchmarks and challenge models contain short task chains [3]–[5]; our evaluation also includes longer synthetic task chains. All compared methods use the same CECs for each workload and task-chain length. These synthetic workloads are not intended to reflect the prevalence of period relations in deployed systems. We exclude $N = 2$, for which Corollary 4 gives optimal phases and the minimum E2E latency from a single GCD computation.

a) Application-Derived CECs: We evaluate five CECs derived from previously published automotive, avionics, and robotic applications [5], [7], [9]–[12]. We preserve the reported task periods and functional dependencies while adopting the standard LET model and independent task phasing assumed in our system model. The caption of Fig. 1 lists the task periods in CEC order. Figure 1(a) illustrates how phase vectors affect their E2E latency.

b) Semi-harmonic: The semi-harmonic workload includes both max-harmonic and $(2, k)$-max-harmonic CECs. Motivated by engine-control benchmark period relations used in prior CEC studies [15], [16], [21], Günzel and Becker defined max-harmonic and $(2, k)$-max-harmonic period sets [7]. A max-harmonic CEC has every period dividing its largest period, so $H_{1:N} = T_{\widehat{p}}$ and $M_{1:N} = 1$. For a $(2, k)$-max-harmonic CEC, the hyperperiod is twice the largest period and k times the second-largest distinct period; every smaller distinct period divides both, giving $M_{1:N} = 2$.

For this workload, we independently sample each task period with replacement from the engine-control benchmark set $\{1, 2, 5, 10, 20, 50, 100, 200, 1000\}$ ms [3]. Every task chain

sampled from this set is max-harmonic unless its two largest distinct periods are $(5, 2)$ or $(50, 20)$ ms, in which case it is $(2, 5)$-max-harmonic. Each task-chain length contains 500 CECs: 468–493 are max-harmonic and the remaining 7–32 are $(2, 5)$-max-harmonic.

c) (x, y)-hyperperiod: Günzel and Becker defined $(2, k)$-max-harmonic periods from the engine-control benchmark [7]. Motivated by the wider range of task periods in reported application examples [4], [5], [12], we define a broader (x, y)-hyperperiod workload. Let $T_{\max,1} > T_{\max,2}$ denote the two largest distinct periods. A CEC belongs to this workload if, for coprime integers $2 \leq x < y$,

$$x \cdot T_{\max,1} = y \cdot T_{\max,2} = H_{1:N}. \tag{13}$$

For example, the task-period vector $(30, 80, 15, 5, 10)$ ms has $H_{1:N} = 240$ ms and satisfies Eq. (13) with $(x, y) = (3, 8)$, although 15 ms does not divide 80 ms. The special case $(x, y) = (2, k)$ differs from $(2, k)$-max-harmonic CECs because smaller distinct periods need not divide both $T_{\max,1}$ and $T_{\max,2}$.

Because $T_{\max,1} = T_{\widehat{p}}$, Eq. (13) gives $M_{1:N} = H_{1:N}/T_{\widehat{p}} = x$. To evaluate a range of x, we sample x uniformly from $\{3, \ldots, 20\}$ and y uniformly from the first ten integers greater than x that are coprime to x. We construct task chains satisfying this workload definition, with the largest period capped at 500 ms, and uniformly permute the task periods. Each task-chain length contains 500 CECs, with 148–189 CECs in each of the ranges $x = 3$–8, 9–14, and 15–20.

d) Randomly sampled: This workload evaluates task chains without an imposed harmonic or coprime period relation. $M_{1:N} = H_{1:N}/T_{\widehat{p}}$ and the number of candidate phases vary across CECs of the same task-chain length. Across the 500 CECs at each length, the minimum, median, and maximum $M_{1:N}$ are 1, 14, and 4,028 at $N = 3$, and 2, 4,666.5, and $2.50 \cdot 10^8$ at $N = 8$, respectively.

For this workload, we independently sample each integer task period with replacement from $[2, 100]$ ms. Each value t has probability proportional to $1/t$. The bounded period range retains both small and large hyperperiods while permitting evaluation through $N = 8$.

C. Comparison with Prior Methods

Figure 1(b) and Fig. 6 report execution times for the application-derived and synthetic workloads, respectively; Table III summarizes the synthetic results at $N = 5$. Medians and P99 values (the 495th of 500 values) are computed per synthetic workload and task-chain length, with execution-time statistics reported only for methods measured on all 500 CECs. For every CEC evaluated by at least two methods, all evaluated methods returned the same minimum E2E latency.

Proposed evaluates orders of magnitude fewer phase vectors than Wang'26. For the application-derived CECs, it evaluates at most five, whereas Wang'26 evaluates $1.125 \cdot 10^3$ to $9.6 \cdot 10^9$, with execution-time speedups of 257 to $9.97 \cdot 10^8$. At $N = 8$, the P99 evaluated phase vector counts are $4 \cdot 10^{13}$, $1.92 \cdot 10^{13}$, and $5.44 \cdot 10^5$ for Wang'26 versus 1, 11,250, and 1,296 for Proposed in the semi-harmonic, (x, y)-hyperperiod, and randomly sampled workloads, respectively. Wang'26 counts are computed analytically.

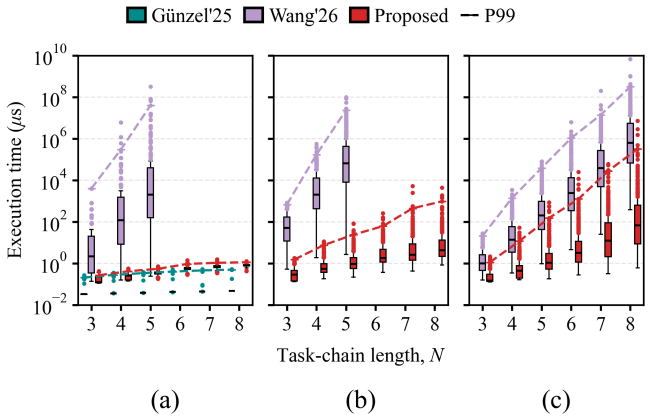

Fig. 6. Algorithm execution times over 500 CECs per task-chain length for (a) semi-harmonic, (b) (x, y)-hyperperiod, and (c) randomly sampled workloads.

The workload-dependent gains are consistent with restricting each extension from g_d integer phases to at most $\min(g_d/\gamma_d, M_{d-1})$ candidates in Eq. (12). Over $N = 3, \ldots, 5$, where Wang'26 was measured in all three workloads, P99 speedups are largest for the semi-harmonic workload, where $M_{1:N} \leq 2$, rising from $1.49 \cdot 10^4$ to $7.32 \cdot 10^7$; for (x, y)-hyperperiod, where $M_{1:N} = x \leq 20$, they rise from 446 to $9.90 \cdot 10^5$. In the randomly sampled workload, bounded periods limit g_d, whereas M_d can be large; P99 speedups range from 18.4 to 983 across $N = 3, \ldots, 8$. The execution time of Proposed also grows with M_d, the number of endpoint pairs in each state: at $N = 8$, Proposed's P99 execution times are 1.14 μs, 965.07 μs, and $3.22 \cdot 10^5$ μs for the three workloads, respectively. For the semi-harmonic workload, Proposed's P99 execution time is 1.29 to 2.33 times that of the closed-form Günzel'25 across $N = 3, \ldots, 8$.

D. Ablation Study

Figure 7 compares the four variants of Proposed, using Proposed without SA and LBP as the execution-time reference for each CEC. Each point for a synthetic workload is the geometric mean of per-CEC ratios over 500 CECs for each task-chain length. LBP can reduce the evaluated phase vector count by pruning branches; SA reduces intermediate work on singleton extensions without changing that count.

SA provides a larger execution-time reduction than LBP in the semi-harmonic workload. LBP reduces the evaluated phase vector count for only 6 of the 3,000 CECs across $N = 3, \ldots, 8$. SA alone yields ratios of 0.601–0.737 across $N = 3, \ldots, 8$. In Proposed without LBP, path-level SA is triggered 356 times at $N = 3$ and 301 times at $N = 8$, including 356 and 6 root completions, respectively. The overall increase in execution-time ratios coincides with fewer root completions. Even when $M_d = 1$ at every depth guarantees singleton candidate phase sets, path-level SA does not necessarily complete the search at the root: it requires the divisibility condition to hold at every extension. SA alone also provides a larger execution-time reduction than LBP alone for AEBS

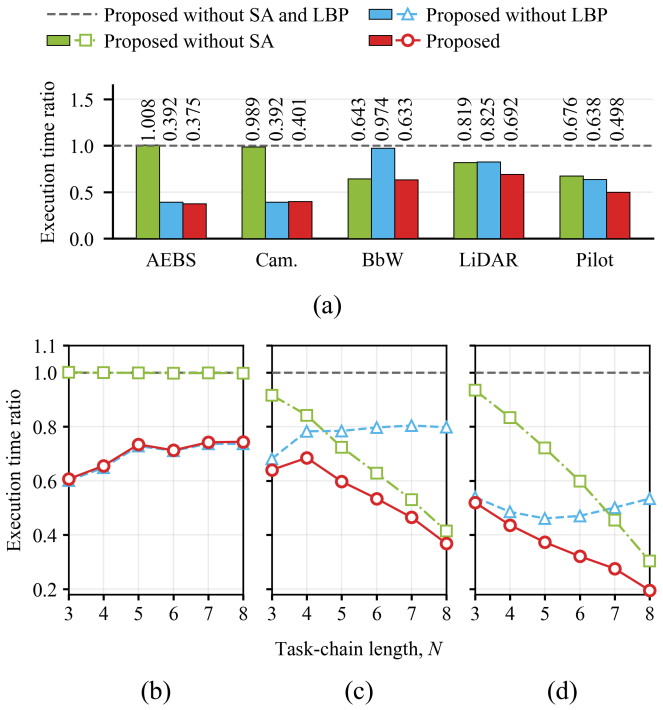

Fig. 7. Algorithm execution times normalized to Proposed without SA and LBP: (a) per-CEC ratios for the application-derived CECs; (b) semi-harmonic, (c) (x, y)-hyperperiod, and (d) randomly sampled workloads. Each point in (b)–(d) is the geometric mean of per-CEC ratios over all 500 CECs.

and camera-based control, with ratios of approximately 0.392 for both CECs.

LBP becomes more effective than SA at larger task-chain lengths in the (x, y)-hyperperiod and randomly sampled workloads. The ratios for Proposed without SA fall from 0.917 and 0.934 at $N = 3$ to 0.415 and 0.304 at $N = 8$, respectively. At $N = 8$, LBP reduces the evaluated phase vector count for 418 and 424 of 500 CECs, respectively. For the CECs with the largest counts without LBP, these decrease from 231,040 to 6,998 and from 15,876 to 6. LBP also reduces counts from 12 to 5 for Brake-by-Wire, 3 to 1 for LiDAR-based control, and 6 to 3 for pilot request handling; it provides the larger execution-time reduction for Brake-by-Wire.

SA's additional benefit with LBP differs between the two workloads. The ratio of Proposed to Proposed without SA rises from 0.699 at $N = 3$ to 0.888 at $N = 8$ in the (x, y)-hyperperiod workload, indicating diminishing SA benefit. This benefit persists in the randomly sampled workload, with a ratio of 0.644 at $N = 8$. Together, they yield ratios of 0.368 and 0.196 relative to the reference at $N = 8$ for these two workloads, respectively.

Aggregate gains do not extend to every CEC. Across $N = 3, \ldots, 8$, Proposed is slower than the reference for 115, 188, and 136 of the 3,000 CECs in the semi-harmonic, (x, y)-hyperperiod, and randomly sampled workloads, respectively. The maximum per-CEC ratios are 1.702 (0.560 versus 0.329 μs), 1.571 (0.443 versus 0.282 μs), and 1.576 (1.417 versus 0.899 μs), comparing Proposed with the reference. In all three cases, Proposed takes less than 1.5 μs, suggesting that SA and LBP check costs can be comparable to search costs.

VII. Conclusion

We presented a method for minimizing the E2E latency of LET-based CECs by optimizing phase vectors, without harmonicity assumptions. The proposed branch-and-bound algorithm searches finite candidate phase sets derived from zero-wait alignment to obtain a globally optimal phase vector. Evaluation across four workloads demonstrated substantial reductions in algorithm execution time, including an execution-time speedup of $9.97 \cdot 10^8$ relative to the enumeration-based method for the pilot request handling CEC in Fig. 1(b). The ablation study showed that singleton acceleration was effective in the semi-harmonic workload, while lower-bound pruning was effective for longer task chains in the other synthetic workloads. Future work will extend the proposed method to jointly optimize phase vectors for multiple CECs that share tasks. The phase of a shared task affects the latencies of multiple CECs, so independently optimal phase vectors may assign incompatible phases to that task.

Acknowledgment

The authors used OpenAI Codex for English editing and assistance with benchmarking scripts for Section VI, and Anthropic Claude for restructuring suggestions and draft figure captions and sentences; the authors revised and verified the AI-assisted material.

References

[1] AUTOSAR, *Specification of Timing Extensions*, AUTOSAR CP R22-11, Document 411, Nov. 2022. [Online]. Available: https://www.autosar.org/fileadmin/standards/R22-11/CP/AUTOSAR_TPS_TimingExtensions.pdf

[2] S. S. Craciunas, M. Becker, and P. Pop, "Automotive cause-effect chains in the wild: A real-world industrial benchmark and generator," Zenodo, ver. 2.0, Oct. 2026, presented at the 15th Int. Real-Time Syst. Open Problems Seminar (RTSOPS). [Online]. Available: https://doi.org/10.5281/zenodo.23336031

[3] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," presented at the 6th Int. Workshop Anal. Tools Methodologies Embedded Real-Time Syst. (WATERS), Lund, Sweden, Jul. 7, 2015.

[4] PerceptIn, "RTSS 2021 Industry Challenge," 2021. [Online]. Available: https://2021.rtss.org/wp-content/uploads/2021/06/RTSS2021-Industry-Challenge-v2.pdf

[5] A. Hamann *et al.*, "WATERS industrial challenge 2019," presented at the 10th Int. Workshop Anal. Tools Methodologies Embedded Real-Time Syst. (WATERS), Stuttgart, Germany, Jul. 9, 2019. [Online]. Available: https://www.ecrts.org/forum/download/WATERS_Industrial_Challenge_2019.pdf

[6] M. Günzel, H. Teper, G. von der Brüggen, and J.-J. Chen, "End-to-end latency of cause-effect chains: A tutorial," *ACM Trans. Embedded Comput. Syst.*, vol. 24, no. 1, Jan. 2025, Art. no. 22.

[7] M. Günzel and M. Becker, "Optimal task phasing for end-to-end latency in harmonic and semi-harmonic automotive systems," in *Proc. IEEE 31st Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2025, pp. 164–176.

[8] M. Günzel, H. Teper, K.-H. Chen, G. von der Brüggen, and J.-J. Chen, "On the equivalence of maximum reaction time and maximum data age for cause-effect chains," in *Proc. 35th Euromicro Conf. Real-Time Syst. (ECRTS)*, ser. LIPIcs, vol. 262. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Jul. 2023, pp. 10:1–10:22. [Online]. Available: https://doi.org/10.4230/LIPIcs.ECRTS.2023.10

[9] S. Wang *et al.*, "Optimizing logical execution time model for both determinism and low latency," in *Proc. IEEE 30th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2024, pp. 135–148.

[10] M. Becker and D. Casini, "The MATERIAL framework: Modeling and AuTomatic code Generation of Edge Real-TIme AppLications under the QNX RTOS," *J. Syst. Archit.*, vol. 154, Sep. 2024, Art. no. 103219.

[11] M. Günzel, M. Becker, and D. Casini, "Shape-aware analysis of end-to-end latency under LET," in *Proc. IEEE 32nd Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2026, pp. 82–95.

[12] M. Lauer, F. Boniol, C. Pagetti, and J. Ermont, "End-to-end latency and temporal consistency analysis in networked real-time systems," *Int. J. Crit. Comput.-Based Syst.*, vol. 5, no. 3/4, pp. 172–196, 2014.

[13] T. A. Henzinger, B. Horowitz, and C. M. Kirsch, "Giotto: A time-triggered language for embedded programming," *Proc. IEEE*, vol. 91, no. 1, pp. 84–99, Jan. 2003.

[14] C. M. Kirsch and A. Sokolova, "The logical execution time paradigm," in *Advances in Real-Time Systems*, S. Chakraborty and J. Eberspächer, Eds. Berlin, Germany: Springer, 2012, pp. 103–120.

[15] S. Wang, E. Bini, Q. Deng, and M. Maggio, "The zeroLET task model and its application to offset design space exploration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2026, to be published, doi: 10.1109/TCAD.2026.3725519.

[16] J. Martinez, I. Sañudo, and M. Bertogna, "Analytical characterization of end-to-end communication delays with logical execution time," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2244–2254, Nov. 2018.

[17] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson, "A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics," presented at the Workshop Compositional Theory Technol. Real-Time Embedded Syst. (CRTS), Barcelona, Spain, Nov. 30, 2008. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1003533/FULLTEXT01.pdf

[18] M. Becker and S. Mubeen, "Timing analysis driven design-space exploration of cause-effect chains in automotive systems," in *Proc. IECON—44th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2018, pp. 4090–4095.

[19] M. Günzel, K.-H. Chen, N. Ueter, G. von der Brüggen, M. Dürr, and J.-J. Chen, "Timing analysis of asynchronized distributed cause-effect chains," in *Proc. IEEE 27th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2021, pp. 40–52.

[20] J. Martinez, I. Sañudo, and M. Bertogna, "End-to-end latency characterization of task communication models for automotive systems," *Real-Time Syst.*, vol. 56, no. 3, pp. 315–347, Jul. 2020.

[21] Y. Tang, X. Jiang, N. Guan, D. Ji, X. Luo, and W. Yi, "Comparing communication paradigms in cause-effect chains," *IEEE Trans. Comput.*, vol. 72, no. 1, pp. 82–96, Jan. 2023.

[22] A. Davare, Q. Zhu, M. Di Natale, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli, "Period optimization for hard real-time distributed automotive systems," in *Proc. 44th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2007, pp. 278–283.

[23] J. Schlatow, M. Möstl, S. Tobuschat, T. Ishigooka, and R. Ernst, "Data-age analysis and optimisation for cause-effect chains in automotive control systems," in *Proc. IEEE 13th Int. Symp. Ind. Embedded Syst. (SIES)*, Jun. 2018, pp. 1–9.

[24] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte, "Synthesizing job-level dependencies for automotive multi-rate effect chains," in *Proc. IEEE 22nd Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2016, pp. 159–169.

[25] M. Verucchi, M. Theile, M. Caccamo, and M. Bertogna, "Latency-aware generation of single-rate DAGs from multi-rate task sets," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2020, pp. 226–238.

[26] M. T. Ali Ahmad, J. Pestana, and M. Baunach, "Packet-based data latency and schedulability optimization in directed acyclic graphs of multirate automotive cause–effect chains," *J. Syst. Archit.*, vol. 179, Oct. 2026, Art. no. 103909.

[27] F. Paladino, A. Biondi, E. Bini, and P. Pazzaglia, "Optimizing per-core priorities to minimize end-to-end latencies," in *Proc. 36th Euromicro Conf. Real-Time Syst. (ECRTS)*, ser. LIPIcs, vol. 298. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Jul. 2024, pp. 6:1–6:25. [Online]. Available: https://doi.org/10.4230/LIPIcs.ECRTS.2024.6

[28] L. Maia and G. Fohler, "Decreasing utilization of systems with multi-rate cause-effect chains while reducing end-to-end latencies," in *Proc. 28th Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2025, pp. 1–10.

[29] L. Maia, M. Günzel, and G. Fohler, "Safe reconfiguration of LET communication intervals to reduce end-to-end latencies," presented at the 33rd Int. Conf. Real-Time Netw. Syst. (RTNS), Pisa, Italy, Nov. 5–7, 2025. [Online]. Available: https://daes.cs.tu-dortmund.de/storages/daes-cs/r/publications/preprint_MaiaGF25.pdf



Jongwon Lee is currently pursuing the M.S. degree with the Department of Future Automotive Mobility, Seoul National University, Seoul, Republic of Korea. His research interests include software-defined vehicles, task scheduling, and embedded and real-time systems design.



Namcheol Lee is currently pursuing the Ph.D. degree in electrical and computer engineering at Seoul National University, Seoul, Republic of Korea. His research interests include software-defined vehicles, on-device AI optimization, and autonomous driving.



Saehwa Kim (Member, IEEE) received her B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Korea, in 1997, 2000, and 2006, respectively. In 1997 and 2006, she was a researcher in the Automation and Systems Research Institute, Korea. In 2007 and 2008, she was a senior research engineer at LG Electronics, Ltd. She is currently a professor at the Department of AI Robotics, Hankuk University of Foreign Studies, Korea. Her research interests include real-time on-device AI systems, mixed-criticality systems, and cloud-native computing.



Seongsoo Hong (Senior Member, IEEE) received the B.S. and M.S. degrees in computer engineering from Seoul National University, Seoul, Republic of Korea, in 1986 and 1988, respectively, and the Ph.D. degree in computer science from the University of Maryland, College Park, MD, USA, in 1994. He is currently a Professor with the Department of Electrical and Computer Engineering, Seoul National University. His current research interests include on-device AI optimization, software-defined vehicles, task scheduling and resource management in the Linux kernel, embedded and real-time systems design, and real-time operating systems. He is a member of the Advisory Committee of IEEE RTCSA. He served as the General Co-Chair of IEEE RTCSA 2006 and CASES 2006 and the Program Committee Co-Chair of IEEE RTAS 2005, RTCSA 2003, IEEE ISORC 2002, and ACM LCTES 2001. He has served on numerous program committees, including IEEE RTSS and ACM OOPSLA. He is a Fellow of the National Academy of Engineering of Korea and a Senior Member of ACM.