

GLEAM: A Graph-Learning Enhanced Adaptive Metaheuristic for Power-Aware Scheduling on Heterogeneous Cyber-Physical Systems

Amir Hossein Ansari^{*✉}, Mohsen Ansari^{*†✉}, Sepideh Safari^{†✉}, Alireza Ejlali^{*✉}, and Jörg Henkel^{‡✉}

^{*}Sharif University of Technology, {amir.ansari79, ansari, ejlali}@sharif.edu, Tehran, Iran

[†]Institute for Research in Fundamental Sciences, {mansari, sepideh.safari}@ipm.ir, Tehran, Iran

[‡]Karlsruhe Institute of Technology, henkel@kit.edu, Karlsruhe, Germany

Abstract—The increasing complexity of embedded and Cyber-Physical Systems (CPS) has accelerated the adoption of heterogeneous multi-core architectures, which combine performance and energy efficiency. However, scheduling dependent tasks on such platforms introduces significant challenges due to strict real-time constraints, high energy consumption, and the NP-hard nature of task mapping. This paper proposes a novel hybrid scheduling framework to jointly optimize energy efficiency and timeliness for Directed Acyclic Graph (DAG) applications. The framework operates in three tiers: first, a Genetic Algorithm (GA) performs a global search to determine near-optimal task-to-core mappings; second, a Dynamic Voltage and Frequency Scaling (DVFS) manager is integrated into the GA’s fitness function to accurately capture energy-performance trade-offs; and third, a Graph Neural Network (GNN) is trained to imitate the GA+DVFS policy, enabling fast and high-quality online scheduling decisions. Experimental results demonstrate that the proposed approach achieves a balanced trade-off between power consumption and deadline satisfaction, while the GNN significantly accelerates scheduling without compromising solution quality. Our GLEAM method reduced energy consumption on average by 49.08% and improved the makespan on average by 27.03% compared to baseline methods.

Index Terms—Cyber-Physical Systems, Task Scheduling, Genetic Algorithm, Graph Neural Networks, Energy Efficiency.

I. INTRODUCTION

With the rapid growth of Cyber-Physical Systems (CPSs), computing platforms have become central to modern industries, ranging from autonomous vehicles and industrial automation to healthcare and the Internet of Things (IoT) [1]–[4]. These systems tightly couple computational processes with the physical environment, requiring them to meet strict performance, energy consumption, reliability, and timing constraints [1], [5]–[7]. To cope with the rising computational demands of real-time applications, designers increasingly rely on heterogeneous multicore System-on-Chip (SoC) platforms, where high-performance cores coexist with low-power cores [4], [5], [8]–[11]. Such architectures offer a promising balance between performance and energy efficiency [3], [5], [7], [11], [12].

Despite these benefits, heterogeneous platforms introduce new challenges for system designers. One key challenge is the trade-off between energy consumption and timing guarantees [5]–[7], [10]–[14]. Real-time CPSs must ensure that each task is completed before its deadline, as missing deadlines can cause instability or even catastrophic system failures [3],

[13], [15]–[17]. However, operating cores at high frequencies (to guarantee timing constraints) leads to a significant increase in power consumption, following the well-known cubic relationship between frequency and dynamic power. Another critical challenge is computational load balancing [5], [6], [14]. Applications are often represented as Directed Acyclic Graphs (DAGs), where tasks have inter-dependencies that dictate strict execution orders [5], [8], [10], [16], [18]. Efficient mapping and scheduling must therefore map and execute tasks across heterogeneous cores while maintaining power and timing constraints, minimizing the makespan, and reducing overall energy consumption [1], [3], [10], [19].

These challenges make scheduling on heterogeneous platforms an NP-hard problem [3], [4], [19]. The vast combinatorial search space of task-to-core mappings, combined with the dynamic voltage and frequency scaling (DVFS) technique [3], [7], [17], [19], [20], renders it computationally infeasible to derive optimal schedules in real-time [3], [7], [18], [19]. Existing heuristic and meta-heuristic approaches partially address these issues but often suffer from limited scalability or high overhead, making them impractical for dynamic workloads [3], [7], [19], [20].

A. Motivational Example

To further illustrate the challenges discussed above, we present a motivational example from a real-world domain, autonomous driving, where these issues are particularly critical. The vehicle’s onboard computer is built on a heterogeneous SoC, and its primary application can be modeled as a DAG in which tasks such as perception, sensor fusion, path planning, and actuator control are tightly interdependent [21]–[23]. The core challenge is to map and schedule this DAG in a way that reduces energy consumption (to extend the vehicle’s range) while strictly meeting all task deadlines (to ensure safety).

This trade-off is not merely theoretical. To illustrate its severity and the clear need for an intelligent solution, we present results from 12 representative configurations, scaling from small to large DAGs on 4, 8, 16, and 32-core systems. Fig. 1 compares our approach with a spectrum of scheduling strategies.

On the one hand, performance-centric heuristics like HEFT [4], [24] consistently achieve the lowest makespan, but at an unsustainable energy cost. For example, in the largest 32-

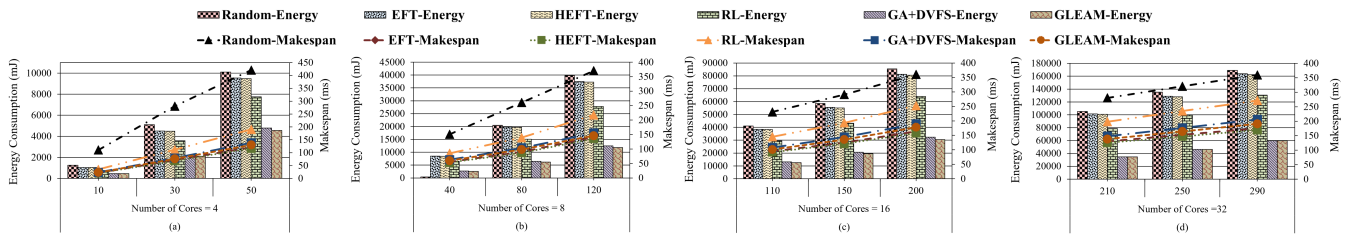


Fig. 1: Motivational comparison of makespan and energy between baseline schedulers and our proposed method (GLEAM) for 12 different configurations on a) 4-core, b) 8-core, c) 16-core, and d) 32-core systems.

core configuration, HEFT consumes over 162,000 (mJ) of energy. Other metaheuristics, such as reinforcement learning-based schedulers, offer only partial compromises, still consuming more than twice the energy of our proposed method (GLEAM). These results clearly demonstrate that conventional heuristic and metaheuristic methods are unable to adequately address the dual requirements of performance and energy efficiency in safety-critical systems.

In contrast, our power-aware expert scheduler (GA+DVFS) demonstrates the possibility of striking this balance. Crucially, our GLEAM method successfully learns this complex policy, delivering perfect results. This highlights the critical need for an intelligent and power-aware scheduler that can navigate this complex trade-off. However, while the GA+DVFS scheduler generates high-quality solutions, it is computationally expensive and unsuitable for deploying real-time applications, such as autonomous driving, with execution times stretching into minutes for complex workloads. As shown in Table I, its execution time can extend to several minutes for complex scenarios. This creates the critical gap that our work addresses, namely, the need for high-quality expert scheduling and the high speed required for online decision-making. GLEAM is the only framework that achieves both.

B. Our Contribution

To bridge this critical gap, we propose GLEAM (a Graph-Learning Enhanced Adaptive Metaheuristic for Power-Aware Scheduling). Our framework divides the problem into an offline expert optimization phase and an online inference phase. We first employ a Genetic Algorithm integrated with the DVFS manager to generate near-optimal, power-aware schedules. We then use these GA-DVFS schedules as ground-truth data to train a Graph Neural Network (GNN). The trained GNN learns the complex scheduling policy of the GA-DVFS schedules and can infer a high-quality schedule for a new task graph in a single, rapid forward pass, achieving a speed-up of over 14,000x as shown in Table I. Fig. 2 illustrates the overall structure of the proposed GLEAM framework.

The main contributions of this paper are as follows:

- A hybrid GA-DVFS and GNN framework that combines the solution quality of metaheuristics with the speed of deep learning to reduce energy consumption and improve makespan while maintaining timing constraints.
- We integrate the DVFS technique into the GA’s fitness evaluation, ensuring that the search is guided to manage toward both energy consumption and timing constraints.
- We design and implement a GNN trained to replicate the GA+DVFS policy, enabling rapid scheduling decisions suitable for online scenarios.

The rest of this paper is organized as follows. Section II details the system and energy model. Section III presents our proposed GLEAM framework. Section IV discusses the experimental setup and evaluates performance. Finally, Section V concludes the paper.

II. SYSTEM AND ENERGY MODEL

This section introduces the hardware platform, task, power, and energy models.

A. Hardware Platform Model

The hardware platform is composed of m heterogeneous processing cores, denoted as $\mathcal{C} = \{C_0, C_1, \dots, C_{m-1}\}$ [8], [10], [19]. Each core $C_j \in \mathcal{C}$ is described by a tuple $C_j = \{S_j, V_j, F_j\}$, where S_j represents the relative processing speed of the core. This parameter captures architectural heterogeneity, for example, the difference between high-performance big cores and low-power LITTLE cores [8], [11], [25]. The term V_j denotes the operational voltage. Finally, F_j is the discrete set of operating frequencies available to C_j , supporting DVFS.

B. Application Task Model

The workload of the system is modeled as a Directed Acyclic Graph (DAG), $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{T_0, T_1, \dots, T_{n-1}\}$ is the set of hard real-time tasks, and

Cores	Tasks	GA Time (s)	GNN Time (ms)	Speed-up
<i>4-Core System</i>				
4	10	2.1	1.8	1167x
4	30	9.4	3.2	2938x
4	50	21.3	4.9	4347x
<i>8-Core System</i>				
8	40	18.5	4.8	3854x
8	80	50.8	7.8	6513x
8	120	98.1	11.5	8530x
<i>16-Core System</i>				
16	110	95.8	12.5	7664x
16	150	151.3	15.2	9954x
16	200	251.0	19.8	12677x
<i>32-Core System</i>				
32	210	295.5	25.1	11773x
32	250	404.2	30.1	13428x
32	290	558.4	36.1	15468x

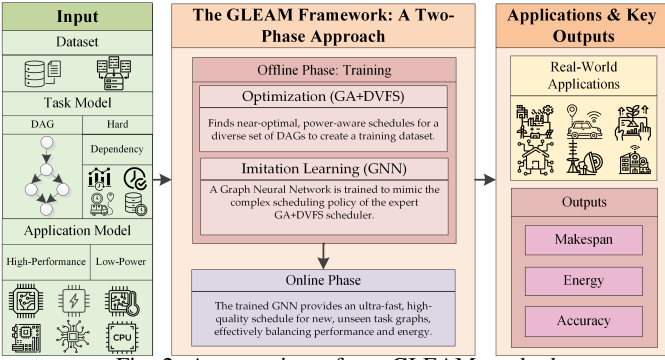


Fig. 2: An overview of our GLEAM method

\mathcal{E} defines the precedence constraints [26], [27]. An edge $(T_i, T_k) \in \mathcal{E}$ indicates that task T_i must finish before task T_k can start. Each task $T_i \in \mathcal{V}$ is defined by the tuple $T_i = \{w_i^{HP}, w_i^{LP}, d_i\}$, where w_i^{HP} denotes the worst-case execution time (WCET) of task T_i when executed on a high-performance (HP) core at its minimum supply voltage and frequency, w_i^{LP} denotes its WCET when executed on a low-power (LP) core at minimum voltage and frequency, and d_i is the absolute deadline for completion. The execution time of a task T_i on core C_j at frequency $f \in F_j$ is given $t_{i,j,f} = \frac{w_i \cdot f_{i,j,f}}{f_i}$.

C. Power and Energy Model

The total power consumption of a core consists of dynamic and static components. The overall instantaneous power can be expressed as $P_j(V_j, f_j) = P_{j,static}(V_j) + P_{j,dynamic}(V_j, f_j)$, where the static term accounts for leakage and the dynamic term depends on voltage, frequency, and task activity. Using the DVFS technique, the frequency-based normalized form becomes [3], [5], [7], [17], [19]:

$$P_j(f_j) = \rho_j P_{j,static}(V_{j,max}) + \rho_j^3 P_{j,dynamic}(V_{j,max}, f_{j,max}), \quad (1)$$

Where $\rho_j = f_j / f_{j,max}$. The energy to execute a task T_i on a core C_j at frequency f_j is $E_{i,j,f} = P_j(f_j) \cdot t_{i,j,f}$. The total system energy is [3], [5], [7], [17], [19], [25]:

$$E_{sys}(\sigma) = \sum_{T_i \in \mathcal{V}} P_{\sigma(i)}(f_{\sigma(i)}) \cdot t_{i,\sigma(i),f_{\sigma(i)}}. \quad (2)$$

III. PROPOSED METHOD

A. Concept Overview

We propose a three-tiered hierarchical framework to address multi-objective scheduling in heterogeneous multi-core systems by separating offline optimization from online decision-making. The main idea is to use computationally intensive methods offline to generate expert schedules and then transfer this knowledge into a lightweight online model for the runtime phase. In the first step, a Genetic Algorithm (GA) explores the task-to-core mapping space. To ensure energy awareness, the GA is combined with a DVFS technique, producing high-quality schedules. In the second step, we overcome the high runtime cost of GA by training a Graph Neural Network

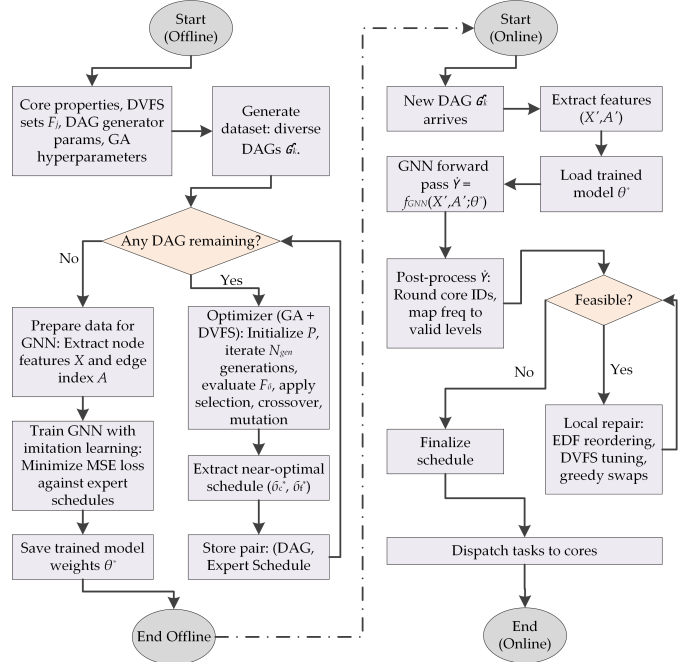


Fig. 3: The proposed GLEAM framework flowchart illustrates its offline and online phases.

(GNN) through supervised imitation learning. The GNN approximates the GA+DVFS policy toward reducing energy consumption and generates expert schedules in a single forward pass with minimal latency, and improves the makespan. This method integrates the accuracy of GA+DVFS in offline optimization with the scalability and speed of GNNs for online scheduling in dynamic environments.

B. Proposed Methodology

To address the multi-objective scheduling problem in heterogeneous multi-core systems, we present a hybrid framework that integrates evolutionary optimization with learning-based inference. The framework consists of two complementary models: a GA operating offline to generate high-quality schedules, and a GNN [28], [29], [29] trained to approximate the GA+DVFS's policy and provide rapid scheduling decisions online. This design uses the accuracy of evolutionary search for offline optimization and the efficiency of graph-based learning for real-time systems.

1) Genetic Algorithm Model

The GA addresses the NP-hard scheduling problem by exploring the combinatorial search space of task-to-core and frequency assignments. Each candidate solution, or chromosome σ , encodes a schedule with two vectors: a core assignment vector $\sigma_c \in \mathbb{Z}^n$ and a frequency assignment vector $\sigma_f \in \mathbb{R}^n$. The fitness function measures the quality of each schedule:

$$F(\sigma) = M(\sigma) + w_E \cdot E_{sys}(\sigma) + \Pi(\sigma), \quad (3)$$

where $M(\sigma)$ is the makespan, $E_{sys}(\sigma)$ is the total system energy, and $\Pi(\sigma)$ penalizes deadline violations. The GA evolves its population iteratively through selection, crossover, and mutation. Competitive selection identifies strong candidates,

Algorithm 1 Hybrid GA-DVFS Scheduling Algorithm

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{C} = \{C_1, \dots, C_m\}$, F_j for each C_j , N_{pop} , N_{gen} , N_{elite} , w_E , W_p

Output: σ^*

```

1:  $P \leftarrow \{\sigma_1, \dots, \sigma_{N_{pop}}\}$  with  $\sigma_k = (\sigma_c^k, \sigma_f^k)$ ,  $\sigma_c^k(i) \in \{1, \dots, m\}$ ,  $\sigma_f^k(i) \in F_{\sigma_c^k(i)}$  // Initialize population with random core/frequency assignments
2:  $\sigma^* \leftarrow \emptyset$ ,  $F^* \leftarrow \infty$ ,  $g \leftarrow 0$  // Initialize best solution and generation counter
3: while  $g < N_{gen}$  do
4:   for  $\sigma \in P$  do
5:      $M(\sigma) \leftarrow \max_{T_i \in \mathcal{V}} c_i$  // Makespan of schedule
6:      $E_{sys}(\sigma) \leftarrow \sum_{T_i \in \mathcal{V}} P_{\sigma_c(i)}(\sigma_f(i)) \cdot t_{i, \sigma_c(i), \sigma_f(i)}$  // System energy using Eq. 2
7:      $\Pi(\sigma) \leftarrow \sum_{T_i \in \mathcal{V}} \max(0, c_i - d_i) \cdot W_p$ 
8:      $F(\sigma) \leftarrow M(\sigma) + w_E \cdot E_{sys}(\sigma) + \Pi(\sigma)$ 
9:   end for
10:  Sort  $P$  by  $F(\sigma)$  in ascending order
11:  if  $F(P[0]) < F^*$  then
12:     $\sigma^* \leftarrow P[0]$ ,  $F^* \leftarrow F(P[0])$ 
13:  end if
14:   $P_{new} \leftarrow \text{SelectElite}(P, N_{elite})$ 
15:  while  $|P_{new}| < N_{pop}$  do
16:     $\sigma_{p1} \leftarrow \text{Tournament}(P)$ 
17:     $\sigma_{p2} \leftarrow \text{Tournament}(P)$ 
18:     $(\sigma_{c1}, \sigma_{c2}) \leftarrow \text{Crossover}(\sigma_{p1}, \sigma_{p2})$ 
19:     $\sigma'_{c1} \leftarrow \text{Mutate}(\sigma_{c1})$ 
20:     $\sigma'_{c2} \leftarrow \text{Mutate}(\sigma_{c2})$ 
21:     $P_{new} \leftarrow P_{new} \cup \{\sigma'_{c1}, \sigma'_{c2}\}$ 
22:  end while
23:   $P \leftarrow P_{new}$ ,  $g \leftarrow g + 1$  // Proceed to next generation
24: end while
25: return  $\sigma^*$ 

```

crossover combines parent schedules, and mutation introduces diversity. Elitism ensures the best chromosomes are preserved, and after several generations, the GA converges to high-quality, energy-aware schedules that act as expert solutions for training the GNN.

2) Graph Neural Network Model

While the GA generates near-optimal solutions, its iterative nature makes it unsuitable for real-time systems. To address this, a GNN trained using imitation learning replicates the GA+DVFS policy. Built on a Graph Convolutional Network (GCN), it processes DAGs directly and predicts per-task core and frequency assignments. Each DAG \mathcal{G} is encoded into a feature matrix X and adjacency matrix A , while the GA+DVFS scheduler σ^* provides supervision. The GNN is trained to minimize the imitation loss:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n (\lambda_c \cdot \text{CE}(\hat{\sigma}_c(i), \sigma_c^*(i)) + \lambda_f \cdot \text{MSE}(\hat{\sigma}_f(i), \sigma_f^*(i))), \quad (4)$$

where $\hat{\sigma}_c(i)$ and $\hat{\sigma}_f(i)$ are the GNN predictions and $\sigma_c^*(i)$, $\sigma_f^*(i)$ are the GA+DVFS scheduler assignments. During inference, the trained GNN predicts a complete schedule in one forward pass,

$$\hat{\sigma} = f_{GNN}(\mathcal{G}; \theta), \quad (5)$$

With cores selected by maximum probability and frequencies discretized to valid levels. This allows the GNN to provide

Algorithm 2 GNN-based Scheduling Algorithm

Input: Task graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, core set $\mathcal{C} = \{C_1, \dots, C_m\}$, frequency sets $\{F_j\}_{j=1}^m$, Expert Scheduler Alg_1 , epochs N_{epochs} .

Output: Predicted schedule $\hat{\sigma} = (\hat{\sigma}_c, \hat{\sigma}_f)$.

```

1: Phase 1: Offline Training
2:  $\sigma^* \leftarrow \text{Alg}_1(\mathcal{G}, \mathcal{C})$  // GA-DVFS schedule
3:  $(X, A) \leftarrow \text{ExtractFeatures}(\mathcal{G})$  //  $X \in \mathbb{R}^{|\mathcal{V}| \times d}$  node features,  $A \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$  adjacency
4:  $Y \leftarrow \text{Encode}(\sigma^*)$  // Target tensor with core and v/f labels
5:  $\theta \leftarrow \text{Initialize}(f_{GNN})$ 
6:  $e \leftarrow 0$ 
7: while  $e < N_{epochs}$  do
8:    $\hat{Y} \leftarrow f_{GNN}(X, A; \theta)$ 
9:    $\mathcal{L}(\theta) \leftarrow \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \|\hat{Y}_i - Y_i\|^2$  // Loss function
10:   $\theta \leftarrow \text{AdamUpdate}(\theta, \nabla_{\theta} \mathcal{L})$ 
11:   $e \leftarrow e + 1$ 
12: end while
13:  $\theta^* \leftarrow \theta$  // Store final trained parameters
14: Phase 2: Online Inference
15: Given new DAG  $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ 
16:  $(X', A') \leftarrow \text{ExtractFeatures}(\mathcal{G}')$  // Preprocess unseen DAG
17:  $\hat{Y}_{raw} \leftarrow f_{GNN}(X', A'; \theta^*)$  // Predict scheduling decisions
18:  $\hat{\sigma} \leftarrow \emptyset$ 
19: for  $i = 1 \rightarrow |\mathcal{V}'|$  do
20:    $\hat{\sigma}_c(i) \leftarrow \text{round}(\hat{Y}_{raw}[i][0])$  // Discretize to nearest core ID
21:    $\hat{\sigma}_f(i) \leftarrow \arg \min_{f \in F_{\hat{\sigma}_c(i)}} |\hat{Y}_{raw}[i][1] - f|$  // Discretize to nearest valid v/f level
22: end for
23: return  $\hat{\sigma}$ 

```

a suitable scheduler that improves the makespan and reduces energy consumption, ensuring scalability and practicality for CPSs. Overall, the methodology combines the accuracy of evolutionary optimization with the efficiency of graph-based learning, enabling high-quality scheduling decisions under performance, energy, and timing constraints. Fig. 3 provides a step-by-step detail of the GLEAM framework, including its offline and online phases.

C. Proposed Algorithms

The proposed method is presented through two algorithms. Algorithm 1 implements the GA+DVFS scheduler by combining a Genetic Algorithm with a DVFS-aware fitness function, while Algorithm 2 describes the GNN-based imitation learning process for the online phase.

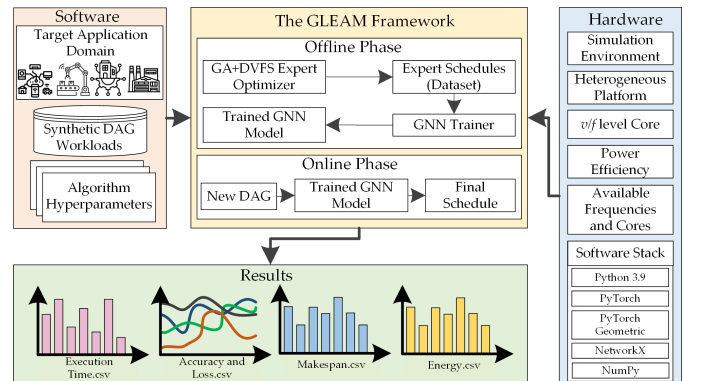


Fig. 4: The GLEAM framework, including its software components, hardware platform, core methodology, and output results.

1) Hybrid GA-DVFS Scheduling Algorithm

The algorithm begins with a randomly initialized population P of N_{pop} chromosomes, where each chromosome σ_k encodes a schedule through a core assignment vector σ_c^k and a frequency assignment vector σ_f^k . The best solution σ^* is initialized as undefined, with fitness F^* set to infinity. In each generation, the fitness $F(\sigma)$ of every chromosome is computed as the weighted sum of makespan, system energy, and a penalty for deadline violations. The population is sorted by fitness, and the best solution is updated if improved. Elitism preserves the top N_{elite} candidates, while the rest of the population is generated through tournament selection, crossover, and mutation. This process iterates for N_{gen} generations, after which the best schedule σ^* is returned. The dominant cost arises from fitness evaluation, which requires $\mathcal{O}(n)$ operations per chromosome, where $n = |\mathcal{V}|$. With N_{pop} chromosomes over N_{gen} generations, the total complexity is $\mathcal{O}(N_{gen} \cdot N_{pop} \cdot n)$.

2) GNN-based Scheduling through Imitation Learning

The GNN-based approach consists of an offline training phase and an online inference phase. In training, expert schedules σ^* are generated by the GA-DVFS algorithm and used as supervision. Each DAG is encoded into a feature matrix X and adjacency matrix A , while the expert schedule is converted into target labels Y . A GNN $f_{GNN}(X, A; \theta)$ is then trained for N_{epochs} epochs using the mean squared

error loss and the Adam optimization algorithm, yielding trained parameters θ^* . During inference, a new DAG using inference, a new DAG \mathcal{G}' is processed into (X', A') and passed through the trained GNN to obtain predictions \hat{Y}_{raw} . These predictions are discretized into valid core and frequency assignments, producing the final schedule $\hat{\sigma}$. Training complexity is $\mathcal{O}(N_{epochs}(|\mathcal{E}|d + |\mathcal{V}|d^2))$, while inference requires only one forward pass with $\mathcal{O}(|\mathcal{E}|d + |\mathcal{V}|d^2)$, making the method suitable for real-time scheduling.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

This section provides a comprehensive evaluation of our proposed method, GLEAM. These tests are designed to evaluate the performance of our system against different baseline algorithms in various ways. Our goal is not only to demonstrate better results in the balance of makespan and energy consumption, but also to provide a suitable solution for heterogeneous multi-core CPSs due to their rapid implementation time. All experiments were conducted in a custom Python 3.9 environment, using NetworkX for DAG generation, NumPy for numerical computations, and PyTorch with PyTorch Geometric for implementing and training the GNN model. To analyze scalability, we generated 36 workload configurations across four platforms: a 4-core system with 10–50 tasks, an 8-core system with 40–120 tasks, a 16-core system with 110–200 tasks, and a 32-core system with 210–290 tasks. These workloads allow a detailed study of performance under

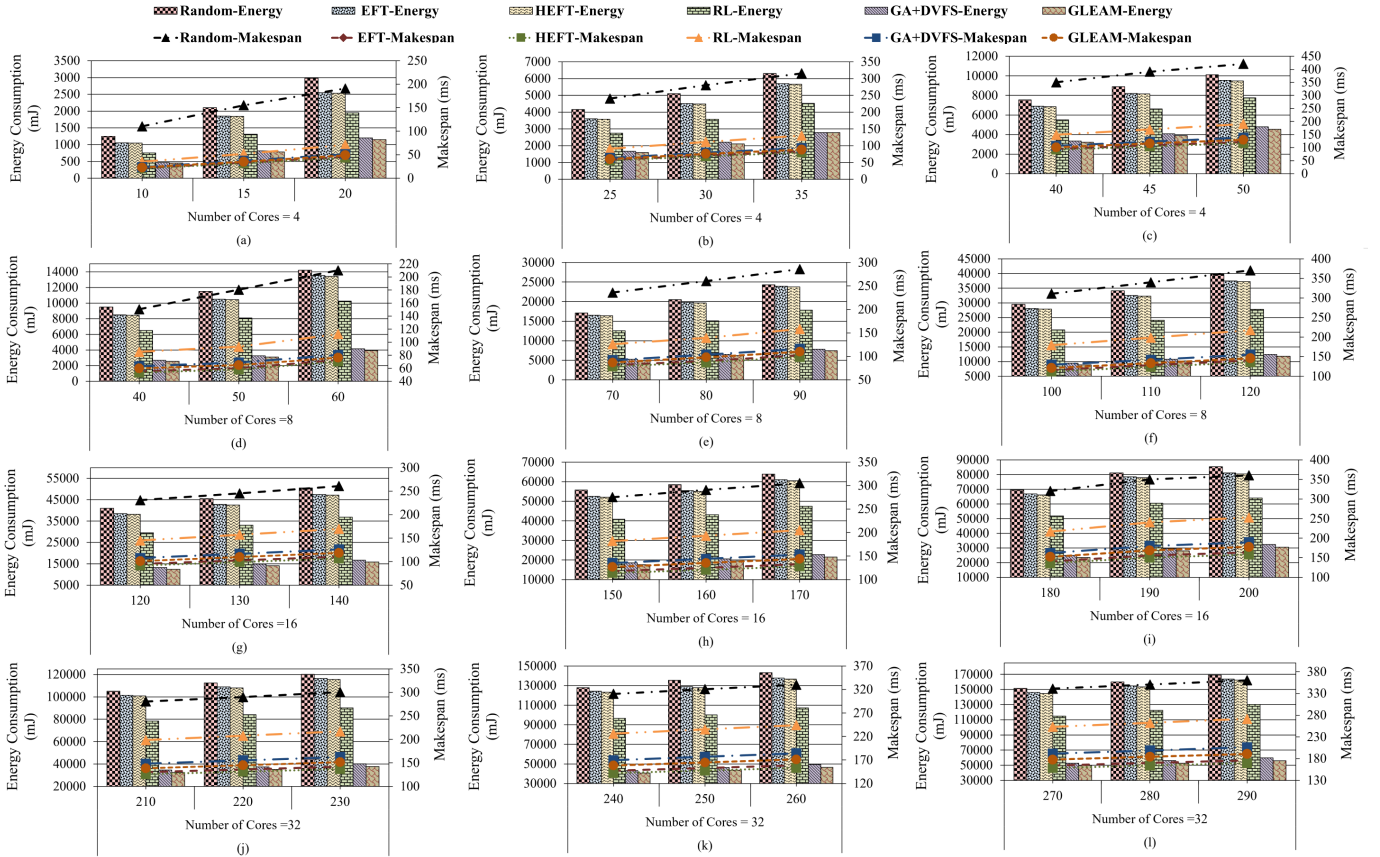


Fig. 5: Final comprehensive comparative analysis of all scheduling algorithms. The figure shows the makespan and total energy consumption for the proposed GLEAM method compared to five other baseline methods across all 36 configurations.

TABLE II: Comprehensive performance metrics of the GLEAM framework. This table compares the execution time of the GA+DVFS and the GNN model’s learning quality (policy accuracy and final loss) across all 36 configurations.

4-Core and 8-Core Systems							16-Core and 32-Core Systems						
Cores	Tasks	GA Time (s)	GNN Time (ms)	Speed-up	Accuracy (%)	Loss (MSE)	Cores	Tasks	GA Time (s)	GNN Time (ms)	Speed-up	Accuracy (%)	Loss (MSE)
4	10	2.1	1.8	1167x	96.7%	0.048	16	110	95.8	12.5	7664x	93.6%	0.031
4	15	3.5	2.1	1667x	96.0%	0.045	16	120	108.2	13.1	8260x	93.3%	0.030
4	20	5.2	2.5	2080x	95.5%	0.046	16	130	121.5	13.9	8741x	93.1%	0.032
4	25	7.1	2.9	2448x	95.2%	0.043	16	140	135.7	14.5	9359x	92.9%	0.031
4	30	9.4	3.2	2938x	94.7%	0.041	16	150	151.3	15.2	9954x	92.7%	0.033
4	35	12.0	3.6	3333x	94.3%	0.042	16	160	168.1	15.9	10572x	92.5%	0.034
4	40	14.8	4.0	3700x	94.0%	0.040	16	170	185.9	16.8	11065x	92.4%	0.035
4	45	17.9	4.4	4068x	93.8%	0.039	16	190	225.4	18.5	12184x	92.1%	0.036
4	50	21.3	4.9	4347x	93.6%	0.038	16	200	251.0	19.8	12677x	91.9%	0.037
8	40	18.5	4.8	3854x	94.5%	0.037	32	210	295.5	25.1	11773x	91.8%	0.038
8	50	25.6	5.5	4655x	94.2%	0.036	32	220	320.8	26.3	12198x	91.6%	0.039
8	60	33.1	6.2	5339x	94.0%	0.035	32	230	347.1	27.5	12622x	91.5%	0.040
8	70	41.5	7.0	5929x	93.9%	0.034	32	240	375.0	28.8	13021x	91.3%	0.041
8	80	50.8	7.8	6513x	93.8%	0.033	32	250	404.2	30.1	13428x	91.1%	0.042
8	90	61.2	8.6	7116x	93.7%	0.032	32	260	435.1	31.5	13813x	91.0%	0.043
8	100	72.5	9.5	7632x	93.5%	0.031	32	270	468.0	33.0	14182x	90.8%	0.044
8	110	84.9	10.4	8163x	93.4%	0.032	32	280	510.3	34.6	14749x	90.6%	0.045
8	120	98.1	11.5	8530x	93.2%	0.033	32	290	558.4	36.1	15468x	90.4%	0.046

increasing task and hardware parallelism. Fig. 4 provides a complete overview of the GLEAM, illustrating its software components, hardware platform, central methodology, and output results.

We compare GLEAM against several baselines and established methods. The random scheduler assigns tasks to cores and frequencies arbitrarily, serving as a naive lower bound. The Earliest Finish Time (EFT) heuristic greedily maps tasks to the core that completes them earliest at maximum frequency, focusing solely on makespan; it has been applied in papers [3], [4], [13]. The Heterogeneous Earliest Finish Time (HEFT) [4], [10], [24] extends EFT to heterogeneous systems by ranking tasks and minimizing their finish times, but like EFT, it ignores energy. The RL-based scheduler represents learning-based heuristics, distributing load across cores with fixed frequencies for basic energy awareness, as in papers [30]–[32]. The GA+DVFS method integrates a genetic algorithm with DVFS to co-optimize makespan and energy, producing high-quality yet computationally expensive solutions. Finally, our proposed GLEAM trains a GNN to imitate GA+DVFS, enabling near-instantaneous, energy-aware scheduling for real-time systems. Functionality is evaluated through five metrics: makespan to measure total completion time, total energy consumption, execution time to evaluate real-time suitability, policy accuracy to quantify fidelity of GLEAM compared to the GA+DVFS optimizer, and the final mean squared error loss.

A. Energy Consumption Analysis

Energy results in Fig. 4 highlight the clear benefits of power-aware scheduling. HEFT, while it is strong in makespan, consumes the most energy—often 200–300% more—due to always running tasks at maximum frequency. In contrast, GLEAM achieves the lowest energy using DVFS to slow non-critical tasks. On average, GLEAM reduces energy consumption by 49.08% and up to 68.73% across platforms, offering near-expert efficiency with minimal overhead. Random algorithm performs worst, EFT and HEFT remain energy-oblivious, and the RL-based method provides only limited savings.

B. Makespan Analysis

The makespan results in Fig. 4 show a clear hierarchy across all platforms and workloads. We evaluated the GLEAM method against several baselines. While HEFT achieves the shortest makespan, GLEAM delivers competitive results, with GLEAM typically within 5–10% of HEFT. This reflects its deliberate strategy of slowing non-critical tasks to save energy while maintaining strong performance. Overall, GLEAM demonstrates improvement across varying workloads (10–290 tasks) and core counts (4, 8, 16, and 32 cores), improving makespan by an average of 27.03% and up to 36.31% in the best cases. Overall, GLEAM demonstrates the best trade-off between makespan and energy consumption.

C. Execution Time Analysis

As shown in Table II, the GA+DVFS method, while producing high-quality schedules, requires significant runtime, ranging from several seconds to several minutes as the workload size increases. In contrast, the GLEAM method generates schedules in only a few milliseconds. Moreover, unlike GA+DVFS, whose runtime grows with problem scale, GLEAM’s inference time remains nearly constant.

D. GNN Model Performance: Accuracy and Loss

The functionality of GLEAM relies on the GNN’s ability to learn the policy accurately. Training results indicate apparent convergence, with the loss (MSE) decreasing rapidly and stabilizing at low values across all workloads, confirming that the model effectively captures scheduling patterns without overfitting. Policy accuracy, defined as the percentage of tasks with identical core and frequency assignments to the policy, consistently exceeds 90%. These results are shown in Table II.

V. CONCLUSION

This paper introduced GLEAM, a novel scheduling framework that integrates GA+DVFS optimization with a GNN-based imitation learning model. By combining the accuracy of evolutionary search with the speed of graph-based inference, GLEAM achieves an effective balance between makespan, energy, and scalability. Experimental results show that GLEAM reduced energy consumption by 49.08% and improved makespan by 27.03% compared to baseline methods.

REFERENCES

- [1] D. Senapati, P. Bhagat, C. Karfa, and A. Sarkar, "Shield: Security-aware scheduling for real-time dags on heterogeneous systems," *ACM Transactions on Cyber-Physical Systems*, vol. 9, no. 1, pp. 1–29, January 2025.
- [2] L. Piardi, P. Leitão, and A. S. de Oliveira, "Fault-tolerance in cyber-physical systems: Literature review and challenges," pp. 29–34, July 2020.
- [3] S. K. Biswas, P. K. Muhuri, and U. K. Roy, "Binary search-based fast scheduling algorithms for reliability-aware energy-efficient task graph scheduling with fault tolerance," *IEEE Transactions on Sustainable Computing*, 2023.
- [4] J. Zhou, M. Zhang, J. Sun, T. Wang, X. Zhou, and S. Hu, "Drheft: Deadline-constrained reliability-aware heft algorithm for real-time heterogeneous mpoc systems," *IEEE Transactions on Reliability*, vol. 71, no. 1, pp. 178–189, 2022.
- [5] A. Roy, H. Aydin, and D. Zhu, "Energy-aware standby-sparing on heterogeneous multicore systems," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.
- [6] B. N. K. Reddy, Y. C. Krishna, P. N. S. Nitish, and S. D. Bharatula, "Optimizing task scheduling in multi-thread real-time systems using augmented particle swarm optimization," in *2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)*, 2024, pp. 666–671.
- [7] Z. Liu, C. Hu, B. Wang, J. Chen, S. Deng, and J. Yu, "A minimizing energy consumption scheme for real-time embedded system based on metaheuristic optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 7, pp. 2276–2289, 2023.
- [8] D. Senapati, K. Rajesh, C. Karfa, and A. Sarkar, "Tmds: Temperature-aware makespan minimizing dag scheduler for heterogeneous distributed systems," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 6, pp. 1–26, 2023.
- [9] G. Xie, Y. Chen, Y. Liu, Y. Wei, R. Li, and K. Li, "Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1629–1640, 2016.
- [10] B. Hu and Z. Cao, "Minimizing resource consumption cost of dag applications with reliability requirement on heterogeneous processor systems," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 12, pp. 7437–7447, 2020.
- [11] M. Lujan, M. McCrary, B. W. Ford, and Z. Zong, "Vulkan vs opengl es: Performance and energy efficiency comparison on the big.little architecture," in *2021 IEEE International Conference on Networking, Architecture and Storage (NAS)*, 2021, pp. 1–8.
- [12] Y. Sharma, S. Moulik, and S. Chakraborty, "Restore: Real-time task scheduling on a temperature aware finfet based multicore," in *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022, pp. 608–611.
- [13] L. Altin, H. Rahmi Topcuoglu, and F. Sadik Gürgen, "Latency-aware multi-objective fog scheduling: Addressing real-time constraints in distributed environments," *IEEE Access*, vol. 12, pp. 62 543–62 557, 2024.
- [14] F. Busato and N. Bombieri, "A performance, power, and energy efficiency analysis of load balancing techniques for gpus," in *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2017, pp. 1–8.
- [15] C. J. Lehmann, L. Bauer, H. Nassar, H. Khdr, and J. Henkel, "Hardware/software co-analysis for worst case execution time bounds," in *2025 Design, Automation Test in Europe Conference (DATE)*, 2025, pp. 1–2.
- [16] Q. He, N. Guan, M. Lv, and Z. Gu, "On the degree of parallelism in real-time scheduling of dag tasks," in *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2023, pp. 1–6.
- [17] A. H. Ansari, M. Ansari, and A. Ejllali, "Taft: Thermal-aware hybrid fault-tolerant technique for multicore embedded systems," *IEEE Embedded Systems Letters*, pp. 1–1, 2024.
- [18] N. Chen, X. Dai, A. Burns, and I. Bate, "A hybrid approach to refine wcrb bounds for dag scheduling using anomaly classification," *IEEE Transactions on Computers*, pp. 1–14, 2025.
- [19] S. Yari-Karin, R. Siyadat-zadeh, M. Ansari, and A. Ejllali, "Passive primary/backup-based scheduling for simultaneous power and reliability management on heterogeneous embedded systems," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 1, pp. 82–93, 2022.
- [20] J. L. C. Hoffmann and A. A. Fröhlich, "Online machine learning for energy-aware multicore real-time embedded systems," *IEEE Transactions on Computers*, vol. 71, no. 2, pp. 493–505, 2022.
- [21] A. Amarnath, S. Pal, H. Kassa, A. Vega, A. Buyuktosunoglu, H. Franke, J.-D. Wellman, R. Dreslinski, and P. Bose, "Hetsched: Quality-of-mission aware scheduling for autonomous vehicle socs," 2022. [Online]. Available: <https://arxiv.org/abs/2203.13396>
- [22] S. Yi, T.-W. Kim, J.-C. Kim, and N. Dutt, "Easyr: Energy-efficient adaptive system reconfiguration for dynamic deadlines in autonomous driving on multicore processors," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 3, pp. 1–29, 2023.
- [23] A. Yano and T. Azumi, "Work-in-progress: Multi-deadline dag scheduling model for autonomous driving systems," in *2024 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, Dec. 2024, p. 451–454. [Online]. Available: <http://dx.doi.org/10.1109/RTSS62706.2024.00049>
- [24] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.
- [25] A. Roy, H. Aydin, and D. Zhu, "Energy-efficient fault tolerance for real-time tasks with precedence constraints on heterogeneous multicore systems," in *Proceedings of the 2019 Tenth International Green and Sustainable Computing Conference (IGSC)*. IEEE, October 2019, pp. 1–8.
- [26] J. Huang, R. Li, X. Jiao, Y. Jiang, and W. Chang, "Dynamic dag scheduling on multiprocessor systems: Reliability, energy, and makespan," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3336–3347, November 2020.
- [27] M. Verucchi, I. S. Olmedo, and M. Bertogna, "A survey on real-time dag scheduling, revisiting the global-partitioned infinity war," *Real-Time Systems*, vol. 59, no. 3, pp. 479–530, August 2023.
- [28] H. Tang and J. Dong, "Solving flexible job-shop scheduling problem with heterogeneous graph neural network based on relation and deep reinforcement learning," *Machines*, vol. 12, no. 8, p. 584, August 2024.
- [29] C.-L. Liu and T.-H. Huang, "Dynamic job-shop scheduling problems using graph neural network and deep reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 11, pp. 6836–6848, November 2023.
- [30] B. Sun, M. Theile, and Z. Qin, "Edge generation scheduling for dag tasks using deep reinforcement learning," *arXiv*, 2023.
- [31] X. Zhao and C. Wu, "Large-scale machine learning cluster scheduling via multi-agent graph reinforcement learning," *arXiv*, 2021.
- [32] C. Peng, M. Wang, J. Liu, L. Mo, and D. Niu, "Energy-efficient real-time dag task scheduling on multicore platform by deep reinforcement learning," *IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 597–602, August 2024.