Energy-Aware Real-Time Scheduling of Sporadic Tasks Using a Hybrid Genetic Algorithm with EDF and DVFS

Ibrahim Gharbi
National School of Computer Science (ENSI), University of Manouba, Tunisia
gharbi.ibrahim@gmail.com (corresponding author)

Hamza Gharsellaoui
National School of Computer Science (ENSI), University of Manouba, Tunisia
hamza.gharsellaoui@ensi-uma.tn

Sadok Bouamama
National School of Computer Science (ENSI), University of Manouba, Tunisia
sadok.bouamama@ensi-uma.tn

Received: 20 June 2025 | Revised: 15 August 2025 | Accepted: 25 August 2025

 DOI: https://doi.org/10.48084/etasr.12849

ABSTRACT

This paper outlines a hybrid scheduling approach for multiprocessor real-time embedded systems. The proposed hybrid scheduler for multiprocessor real-time embedded systems integrates Earliest-Deadline-First (EDF) priorities with Dynamic Voltage and Frequency Scaling (DVFS) inside a Genetic Algorithm (GA). This approach tackles the NP-hard problem of scheduling sporadic tasks while jointly minimizing makespan and energy, avoiding deadline misses, and maintaining load balance. EDF is applied during fitness evaluation to enforce deadline-aware dispatch, and DVFS is co-evolved per task. In a heavy-load scenario with 150 task instances on 4 processors, the proposed GA (EDF=On, DVFS=On) achieved a 66.07 ms makespan and 0.20113 J energy, scheduling 150/150 tasks with zero deadline misses. Compared against the Non-dominated Sorting Genetic Algorithm II (NSGA-II) baseline without EDF/DVFS (makespan 253.17 ms, energy 0.36813 J, 132/150 scheduled), the proposed method reduced makespan by ~74% and energy by ~45%, while eliminating deadline misses. In a lighter 100-task scenario, both methods met all deadlines with comparable makespan/energy. These results highlight the benefit of unifying EDF and DVFS within an evolutionary framework for energy-aware real-time scheduling.

Keywords-Real-time scheduling; sporadic tasks; multiprocessor systems; Genetic Algorithm; EDF; DVFS; energy efficiency

I. INTRODUCTION

Real-time embedded systems play a significant role in modern life, from small smart devices to IoT. Such systems have a vital need for efficient scheduling policies, especially under tight constraints on energy consumption and response time. Scheduling sporadic real-time tasks is an NP-hard problem [1]. Traditional heuristics such as Earliest Deadline First (EDF) or Rate Monotonic (RM) offer simplicity and predictability, but fail to optimize multiple objectives simultaneously. To overcome this limitation, many researchers have explored metaheuristic approaches such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), which allow flexible trade-offs between objectives but often suffer from poor convergence or premature stagnation.

GA-based techniques can quickly converge to high-quality solutions when appropriately guided [2]. The proposed GA-based scheduling algorithm integrates EDF [3], a classic real-time scheduling policy, and DVFS (Dynamic Voltage and Frequency Scaling) [4], which trades off execution speed for energy savings by adjusting processor frequency. This combined approach addresses the need to minimize power consumption and makespan while reducing deadline violations and maintaining load balance. For comparison, the proposed algorithm was compared with a standard multi-objective evolutionary baseline, NSGA-II [5].

A. Related Work

Energy-aware real-time scheduling has been examined from various angles. DVFS is a widely adopted technique to reduce energy consumption by adjusting processor speeds

according to workload [6, 7]. For example, in [8], a DVFS-aware scheduling method for parallel applications achieved significant energy savings. Similarly, in [9], a deep reinforcement learning model (FiDRL) was proposed to dynamically adjust DVFS invocation intervals. In [10], reinforcement learning was applied for CPU frequency scaling in soft real-time systems. However, these approaches focus on DVFS control and do not solve the scheduling of sporadic task sets with hard deadlines. Reinforcement learning-based DVFS methods [10, 11] demonstrate the potential of AI in energy management but do not incorporate classical scheduling heuristics, such as EDF or evolutionary search techniques.

GAs have also been applied to real-time task scheduling to overcome the limitations of fixed-priority or greedy heuristics. A recent study in [12] explored energy-efficient real-time scheduling in edge computing systems using a GA, achieving promising results. Meanwhile, priority-driven heuristics remain important. EDF is one of the most effective scheduling algorithms under dynamic task loads. In [13, 14], the EDF was extended to special contexts, namely time-triggered tasks and self-suspending tasks, respectively. Although EDF ensures predictable scheduling behavior, it does not optimize secondary objectives, such as energy or load balancing, unless combined with additional techniques. This insight has led to hybrid strategies that combine heuristics and optimization. In [15], a recent review of embedded real-time scheduling techniques was presented, emphasizing the benefits of combining priority assignment with task criticality awareness. The trend toward hybridization is evident, yet none of the existing works unifies EDF, GA, and DVFS in a single optimization framework. For example, in [16], energy savings were achieved using dynamic voltage scaling but focused solely on uniprocessor scheduling, and in [17], a 60% energy reduction in GPU workloads was achieved using DVFS, although targeting multimedia frame rendering rather than general task scheduling.

In summary, while numerous contributions address either energy efficiency, real-time feasibility, or scheduling optimization, few combine all three aspects (EDF-based scheduling, DVFS-based energy control, and GA-driven search) in one framework. This study fills this gap by integrating EDF and DVFS directly into the evolutionary loop of a GA. This results in an adaptive, power-aware scheduler that respects task deadlines and optimizes makespan and energy consumption concurrently, making it well-suited for real-time embedded environments with sporadic workloads and mixed criticality constraints. The results demonstrate that such integration yields superior outcomes; for example, the proposed GA+EDF+DVFS approach trims schedule length by more than 50% and energy by ~45% versus a standard multi-objective GA, while still meeting all deadlines.

II. METHODOLOGY

A. System Model

Each sporadic task instance is characterized by the 4-tuple:

$$\tau_i = (C_i, a_i, d_i, L_i) \quad (1)$$

where C_i is the Worst-Case Execution Time (WCET), a_i is the arrival time, d_i is the absolute deadline, and $L_i \in \{1,2,3\}$ is the criticality level (1 low, 2 medium, 3 high). Execution time scales inversely with the selected frequency f_i:

$$T_i(f_i) = \frac{C_i}{f_i}, f_i \in [f_{min}, f_{max}] \quad (2)$$

In these experiments, f_{min} was set to 0.7 and f_{max} was set to 1.4. When DVFS is disabled, f_i is fixed to 1.

B. Power and Energy Model

Processor power combines dynamic and static components:

$$P_i(f_i) = \underbrace{\alpha(V_0 + \gamma f_i)^2 f_i}_{\text{dynamic}} + \underbrace{\beta(1 + 0.1x_i)}_{\text{static}} \quad (3)$$

where f_i is the normalized frequency and $x_i \in [-1, 1]$ is a per-instance parameter modeling process/temperature variability. Note that x_i affects only power/leakage, not timing (i.e., not T_i). The same constants were used in the implementation: $V_0 = 0.5$ V, $\gamma = 0.3$, $\alpha = 10^{-3}$, and $\beta = 5 \times 10^{-5}$ W. Therefore, energy per task is given by:

$$E_i = P_i(f_i) \cdot T_i(f_i) \quad (4)$$

C. Optimization Problem

The problem is formulated as a bi-objective optimization:

$$\min_{p,f}[M_s(p, f), E_{tot}(f)] \quad (5)$$

subject to timing and DVFS bounds:

$$a_i + T_i(f_i) \le d_i, \quad i = 1, \dots, n \quad (6)$$

$$f_{min} \le f_i \le f_{max}, \quad i = 1, \dots, n \quad (7)$$

where $M_s = max_{1 \le j \le M} \sum_{i \in \tau_j} T_i(f_i)$ is the makespan across M processors (with τ_j the set of tasks mapped to processor j, and $E_{tot} = \sum_{i=1}^{n} E_i$).

D. Genetic Representation

A chromosome encodes one gene per task instance:

$$gene_i = (p_i, f_i) \quad (8)$$

where p_i is the processor index and f_i is the DVFS setting. This joint encoding lets the GA search processor assignments and frequency choices within the same individual. In the NSGA-II baseline, genes do not include f_i, but each gene encodes only the processor assignment p_i.

III. HYBRID GA–EDF–DVFS FRAMEWORK

A. EDF Integration (Hybrid GA)

Although exact optimality cannot be guaranteed for an NP-hard search space, the hybrid GA inherits standard properties of elitist evolutionary algorithms: (i) with a finite chromosome space, the selection-crossover-mutation process induces an ergodic Markov chain, and (ii) preserving at least one elite individual each generation makes the best-so-far fitness a non-increasing sequence that converges almost surely.

Feasibility is enforced during evaluation (via EDF-based dispatch and deadline checks), and elitism maintains progress without relying on any EDF seeding in initialization.

$$\begin{cases} f_i \sim \mathcal{U}\,(0.7,1.4), & if\ DVFS\ is\ enabled \\ f_i = 1.0, & otherwise \end{cases} \quad (9)$$

- Crossover: Standard two-point crossover. EDF affects only the evaluation (dispatch order), not the crossover mask.
- Evaluation: During fitness computation, each processor's queue is dispatched in EDF order when EDF is enabled. Feasibility is checked implicitly, and missed deadlines incur a penalty.

B. DVFS Integration

Each gene encodes (p_i, f_i). During mutation, both processor and frequency may change with probability $indpb = 0.5$. Processor change is guided by a load/slack score:

$$score(p) = 0.7 \times \frac{load(p)}{max_q\ load(p)} + 0.3 \times \delta$$

$$\delta = \left(1 - \frac{slack(p)}{max_q\ slack(q) + \varepsilon}\right) \quad (10)$$

and the new processor is $arg\,min_p\ score(p)$. Frequency change uses an urgency-modulated uniform step:

$$slack_i = d_i - \frac{c_i}{f_i^{old}}$$

$$u_i = max\left(0.1, \min\left(1.0, 1 - \frac{slack_i}{d_i}\right)\right) \quad (11)$$

$$\Delta f_i \sim \begin{cases} \mathcal{U}(-0.3, -0.1), & if\ u_i < 0.5 \\ \mathcal{U}(-0.15, 0.15), & otherwise \end{cases}$$

$$f_i^{new} = clip\left(f_i^{old} + \Delta f_i, 0.7, 1.4\right) \quad (12)$$

C. Fitness Evaluation and Objective Handling

For each chromosome:

1. Simulate EDF dispatch on all *M* processors.
2. Compute makespan $M_s = max_j \sum_{i \in \tau_j} T_i(f_i)$, total energy $E_{tot} = \sum_i E_i$, and load-balance σ.
3. Aadd a 10 ms penalty to M_s for each missed deadline (none in final solutions).

The single-objective GA minimizes a weighted sum with weights 0.7/0.2/0.1 of (M_s, E_{tot}, σ) when there are misses, and 0.4/0.4/0.2 otherwise, plus penalties $0.01 \times mean((f_i - 1.2)^2)$ and $0.05 \times \sigma,\ if\ \sigma > 0.2$.

For the NSGA-II baseline, DVFS is disabled; all tasks execute at a fixed normalized frequency $f_i = 1$ and NSGA-II uses Pareto ranking over the same objectives evaluated at this fixed frequency.

Algorithm 1 outlines the process from population initialization to final schedule generation in the proposed hybrid GA.

```
Algorithm 1: Hybrid GA-EDF-DVFS Scheduling
Require: T (tasks), G (80 generations),
  P (200 population size),
  M (4 processors), [f_min, f_max]= [0.7,1.4].
Ensure: schedule minimizing M_s, E_tot,
  missed deadlines, and load balance σ.
1. Initialize population with random
   processor assignment and f_i as in (9).
2. Evaluate all individuals with EDF
   dispatch; compute (M_s, E_tot, σ) and apply
   penalties (10 ms per missed deadline to
   minimize M_s energy/load penalties as
   specified).
3. Repeat for g = 1 … G or until early
   stopping:
     • Select (tournament (k=3)).
     • crossover (two-point)
     • Mutation (processor + DVFS):
       o Reassign to processor minimizing
         the load/slack score (10).
       o Update frequency with the urgency-
         modulated step with (11) and (12)
         and clip to [0.7,1.4].
4. Evaluate offspring as in step 2; elitism
   preserves the best-so-far.
5. Return the best individual.
Early Stopping: After 70 generations, stop
if the best fitness improves by less than
0.1% over 5 consecutive generations.
```

D. Theoretical Properties

Although exact optimality cannot be guaranteed for an NP-hard search space, the hybrid GA inherits standard properties of elitist evolutionary algorithms:

1. With a finite chromosome space, the selection-crossover-mutation process induces an ergodic Markov chain.
2. Preserving at least one elite individual each generation makes the best-so-far fitness a non-increasing sequence that converges almost surely.

Feasibility is enforced during evaluation via EDF-based dispatch and deadline checks, and elitism maintains progress without relying on any EDF seeding in initialization.

TABLE I. COMPUTATIONAL COMPLEXITY

Component	Complexity	Description
Initialization	$O(n)$	Random processor assignment
Fitness evaluation	$O(n\log M + n)$	EDF-based simulation ($n\log M$) + DVFS energy and slack calculations (n)
Crossover (EDF-DVFS)	$O(n)$	Two-point crossover
Mutation (DVFS)	$O(n)$	Slack-based urgency evaluation + frequency mutation + clipping
Total per generation	$O(P(n\log M + n))$	P: population size, dominated by fitness evaluation steps

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. Experimental Setup

1) Task Set and Simulation Configuration

The proposed hybrid GA-EDF-DVFS approach was evaluated on a simulated multiprocessor platform with $M = 4$ identical processors. Two loads were considered: a heavy-load scenario with 150 task instances and a lighter-load scenario with 100 instances, both generated from 20 distinct sporadic tasks over a 100-ms horizon. Arrivals are configured so the 150-job case forms a high-utilization stress test (bursty releases), while the 100-job case is more evenly spaced. Task WCETs follow a Weibull distribution to capture realistic variability. Each task has a criticality level $L = \in \{1, 2, 3\}$. In this implementation, static power scales with criticality via the $\beta(1 + 0.1\ L_i)$ term in the power model. Processor frequency is normalized and, when DVFS is enabled, varies within $f \in [0.7, 1.4]$ (representative of 0.7–1.4 GHz in hardware). Power-model parameters $(\alpha, V_{0,}, \gamma, \beta)$ match the methodology section.

2) Genetic Algorithm Parameters (Hybrid GA)

A population of 200 was used, run for 80 generations. Tournament selection (size 3) was applied. Two-point crossover was used with probability 0.9 per parent pair. Mutation is applied with probability 0.4 per offspring and adjusts a subset of task frequencies with urgency-dependent uniform steps (followed by clamping to $[0.7, 1.4]$). If a processor becomes overloaded, one task is migrated to the least-loaded processor. Fitness is computed by EDF simulation on each processor. Makespan M_s, total energy E_{tot}, and load-balance σ, are reported, and a 10 ms penalty is added to M_s per missed deadline.

3) Baseline Comparison (NSGA-II)

For the multi-objective baseline (NSGA-II) [5], DVFS is disabled, and all tasks execute at a fixed frequency $f = 1.0$. The chromosome encodes only processor assignments (no frequency genes). NSGA-II uses Pareto ranking over the same objectives (M_s, E_{tot}, σ), evaluated with the same energy model and missed-deadline penalty, without EDF integration (no deadline-ordering heuristic). The same population (200) and generations (80) are kept, as in the GA, and early stopping is enabled only for NSGA-II (5-generation window, 0.1% tolerance). This baseline isolates the contribution of the co-evolutionary EDF+DVFS design in the proposed GA.

B. Results and Discussion

1) Optimization Evolution

Figures 1 and 2 illustrate GA's convergence behavior for the 150-task scenario. The hybrid GA (EDF+DVFS) rapidly converges to a much lower makespan than any version of NSGA-II. Within ~20 generations, the GA finds schedules around a 70 ms makespan, whereas NSGA-II (without EDF/DVFS) stagnates around 140 ms. Similarly, for energy, the GA quickly drives energy down (to ~0.20 J) while NSGA-II plateaus at a higher energy (~0.26 J). The integration of domain knowledge (EDF ordering and DVFS awareness) clearly improves the search efficiency and outcome.

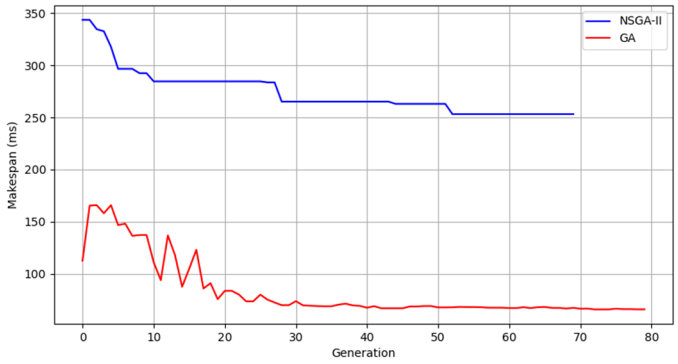

Fig. 1. Makespan evolution over generations.

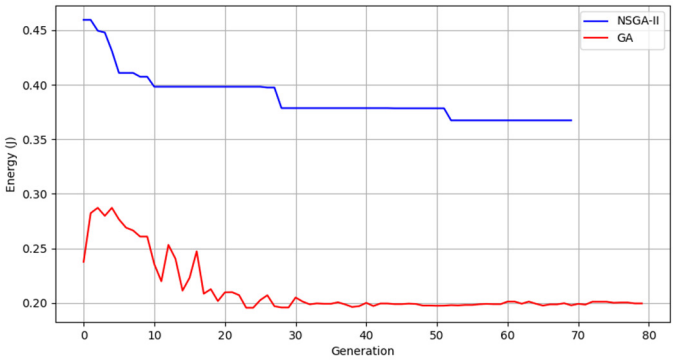

Fig. 2. Energy evolution - GA significantly outperforms NSGA-II.

2) Scheduling Visualization

Figure 3 visualizes one of the final schedules produced by each algorithm for the 150-task case. Using GA (EDF+DVFS), all 150 tasks are scheduled. In the GA schedule, all 150 tasks meet their deadlines (no red deadline-miss markers). The timeline is densely packed: tasks on all four processors run back-to-back with minimal idle gaps, indicating excellent load balancing. The GA schedule effectively utilizes slack by slowing down some tasks (indicated by slightly longer bars for lower frequency tasks) without leaving processors idle.

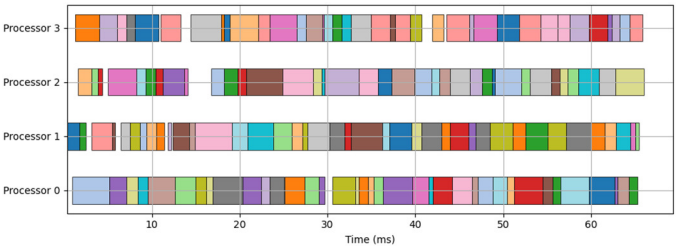

Fig. 3. GA (EDF+DVFS) Gantt chart - 150/150 tasks scheduled.

Figure 4 shows the NSGA-II Gantt chart, where 132/150 tasks are scheduled. The NSGA-II schedule shows several tasks missing deadlines (18 out of 150, marked as incomplete bars) and noticeable idle periods on some processors, even while others are overloaded. This fragmentation and those missed tasks reflect NSGA-II's difficulty in finding a feasible schedule under heavy load without EDF guidance. The GA schedule is both more compact (finishes much earlier) and feasible.

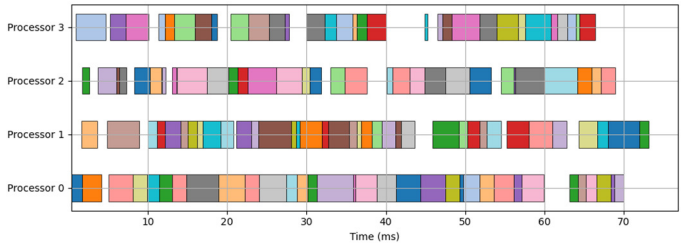

Fig. 4. NSGA-II Gantt chart - 132/150 tasks scheduled.

3) *Frequency Scaling Behavior*

Figure 5 shows the distribution of frequencies used by the proposed GA in the 150-task schedule, showing a balanced scaling across tasks. It can be observed that a range of frequencies is utilized: many tasks run at mid-level frequencies (e.g., 1.0–1.4 GHz) rather than 1.0 GHz. The GA has learned to slow down tasks that are not on the critical path or that have ample slack, thereby saving energy, while still using high frequency for the most urgent tasks. This balanced DVFS usage demonstrates the algorithm's ability to exploit energy-saving opportunities. In contrast, NSGA-II without DVFS would run everything at max frequency (a single spike at f_{max}), wasting potential energy savings, whereas NSGA-II with DVFS but no EDF often misallocates frequency (some tasks might be slowed inappropriately, contributing to deadline misses).

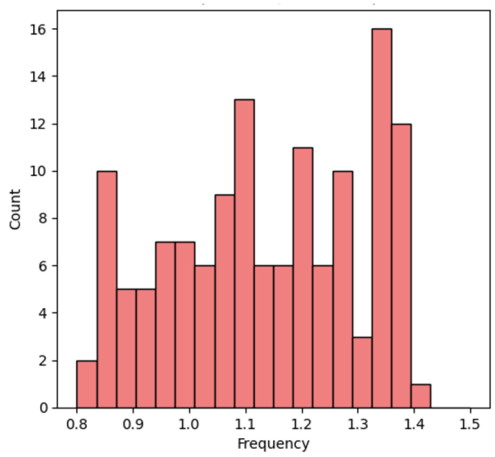

Fig. 5. Frequency distribution in GA (EDF+DVFS). Balanced scaling across tasks.

4) *Quantitative Performance Comparison*

Table I summarizes key performance metrics for various configurations in both scenarios (150 and 100 tasks). The metrics include the schedule makespan, total energy consumption, a load balance indicator (lower is better), and the number of tasks successfully scheduled (out of total).

For 100 tasks, the hybrid GA (EDF+DVFS) achieves a makespan of 60.98 ms, energy of 0.12724 J, zero missed deadlines, and a load balance of 0.17, with early stopping at generation 79. NSGA-II (EDF=Off, DVFS=Off) yields 62.21 ms, 0.12434 J, zero misses, and a balance of 1.67, stopping at generation 70.

TABLE II. PERFORMANCE COMPARISON ACROSS CONFIGURATIONS.

Configuration	Makespan (ms)	Energy (J)	Load balance	Scheduled (tasks)
150 Tasks, adjusted inter-arrival				
NSGA-II (EDF=Off, DVFS=Off)	253.17	0.36813	2.97	132/150
GA (EDF=ON, DVFS=ON)	66.07	0.20113	0.98	150/150
GA (EDF=ON, DVFS=OFF)	69.74	0.18606	0.90	150/150
GA (EDF=OFF, DVFS=ON)	81.06	0.21451	4.07	149/150
GA (EDF=OFF, DVFS=OFF)	164.42	0.28016	4.99	141/150
100 Tasks, adjusted inter-arrival				
NSGA-II (EDF=Off, DVFS=Off)	62.21	0.12434	1.67	100/100
GA (EDF=On, DVFS=On)	60.98	0.12724	0.17	100/100

For 150 tasks, the hybrid GA (EDF=On, DVFS=On) achieves 66.07 ms, 0.20113 J, zero misses, and a balance of 0.98, outperforming NSGA-II (253.17 ms, 0.36813 J, 18 misses, 2.97 balance). Additional configurations for 150 tasks show GA (EDF=On, DVFS=Off) at 69.74 ms, 0.18606 J, zero misses, 0.90 balance, GA (EDF=Off, DVFS=On) at 81.06 ms, 0.21451 J, 1 miss, 4.07 balance, and GA (EDF=Off, DVFS=Off) at 164.42 ms, 0.28016 J, 9 misses, 4.99 balance. In the 150-task run, DVFS-On prioritized latency reduction (66.07 ms vs. 69.74 ms) at a small energy cost relative to the DVFS-Off variant; however, both remain substantially below the NSGA-II baseline energy.

The hybrid GA converges rapidly, reaching 66 ms makespan within 20 generations for 150 tasks, while NSGA-II stagnates around 253 ms. Energy drops to 0.20113 J for GA vs. 0.36813 J for NSGA-II, highlighting the efficacy of EDF ordering and DVFS optimization.

V. CONCLUSION

This study presented a novel hybrid framework for scheduling sporadic real-time tasks on multiprocessors that integrates EDF scheduling heuristics and DVFS-based energy management into a GA. This unified GA-EDF-DVFS approach optimizes multiple objectives simultaneously, achieving substantial reductions in schedule length and energy consumption without missing any task deadlines. Experimental evaluation showed that the hybrid algorithm outperforms a state-of-the-art multi-objective GA (NSGA-II [5]) by a wide margin under heavy load, for example, reducing makespan by ~74% and energy by ~45%, while maintaining feasibility and good load balance. These improvements are possible because the proposed method uniquely exploits the deadline-awareness of EDF and the exploration capability of GA, combined with fine-grained DVFS adjustments for energy efficiency.

The novelty of this work lies in the combination of three strategies that were previously used independently: a priority-based scheduler (EDF) to ensure timing correctness, dynamic frequency scaling to save energy, and evolutionary optimization (GA) to search the vast schedule space. Coupling these addresses the limitations of each, achieving an optimal

balance between feasibility and efficiency. In practical terms, this enables embedded systems to operate with lower energy usage while meeting real-time requirements.

The results of this work were compared with previous ones, such as reinforcement learning-based DVFS methods [9-11] and earlier GA-based schedulers [12], which lack the integrated EDF and DVFS approach, thus underperforming in deadline guarantees and energy savings. In conclusion, the hybrid GA-EDF-DVFS scheduler is a promising solution for energy-aware real-time scheduling in embedded multiprocessor systems. This approach could be extended to various real-time contexts. In future work, we plan to explore online adaptive versions and validate the approach on hardware (e.g., ARM-based multicore boards with DVFS) to measure real energy savings.

REFERENCES

[1] G. C. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, vol. 24. Boston, MA, USA: Springer US, 2011.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[3] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973, https://doi.org/10.1145/321738.321743.

[4] M. Emami, Y. Ghiasi, and N. Jaberi, "Energy-Aware Scheduling using Dynamic Voltage-Frequency Scaling." arXiv, Jun. 10, 2012, https://doi.org/10.48550/arXiv.1206.1984.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002, https://doi.org/10.1109/4235.996017.

[6] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Transactions on Computers*, vol. 53, no. 5, pp. 584–600, Feb. 2004, https://doi.org/10.1109/TC.2004.1275298.

[7] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*, Milwaukee, WI, USA, 1995, pp. 374–382, https://doi.org/10.1109/SFCS.1995.492493.

[8] J. Huang, Y. Chen, L. Xiao, and H. Zeng, "A Dvfs-Weakly Dependent Real-Time Scheduling for Multiple Parallel Applications on Energy-Aware Heterogeneous Systems." SSRN, 2025, https://doi.org/10.2139/ssrn.5090506.

[9] J. Li *et al.*, "FiDRL: Flexible Invocation-Based Deep Reinforcement Learning for DVFS Scheduling in Embedded Systems," *IEEE Transactions on Computers*, vol. 74, no. 1, pp. 71–85, Jan. 2025, https://doi.org/10.1109/TC.2024.3465933.

[10] T. Zhou and M. Lin, "CPU frequency scheduling of real-time applications on embedded devices with temporal encoding-based deep reinforcement learning," *Journal of Systems Architecture*, vol. 142, Sep. 2023, Art. no. 102955, https://doi.org/10.1016/j.sysarc.2023.102955.

[11] A. Yeganeh-Khaksar, M. Ansari, S. Safari, S. Yari-Karin, and A. Ejlali, "Ring-DVFS: Reliability-Aware Reinforcement Learning-Based DVFS for Real-Time Embedded Systems," *IEEE Embedded Systems Letters*, vol. 13, no. 3, pp. 146–149, Sep. 2021, https://doi.org/10.1109/LES.2020.3033187.

[12] H. Hussain *et al.*, "Energy Efficient Real-time Tasks Scheduling on High Performance Edge- Computing Systems using Genetic Algorithm." In Review, Mar. 06, 2023, https://doi.org/10.21203/rs.3.rs-2644571/v1.

[13] A. Finzi, S. S. Craciunas, and M. Boyer, "A Real-time Calculus Approach for Integrating Sporadic Events in Time-triggered Systems." arXiv, 2022, https://doi.org/10.48550/ARXIV.2204.10264.

[14] M. Günzel, K. H. Chen, and J. J. Chen, "EDF-Like Scheduling for Self-Suspending Real-Time Tasks." arXiv, Nov. 18, 2021, https://doi.org/10.48550/arXiv.2111.09725.

[15] J. Yu and X. Lu, "Improvement and Application of Task Scheduling Algorithm for Embedded Real-Time Operating System," in *Proceedings of the World Conference on Intelligent and 3-D Technologies (WCI3DT 2022)*, vol. 323, R. Kountchev, K. Nakamatsu, W. Wang, and R. Kountcheva, Eds. Springer Nature Singapore, 2023, pp. 621–628.

[16] J. J. Chen and C. F. Kuo, "Energy-Efficient Scheduling for Real-Time Systems on Dynamic Voltage Scaling (DVS) Platforms," in *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*, Daegu, Korea, Aug. 2007, pp. 28–38, https://doi.org/10.1109/RTCSA.2007.37.

[17] G. Konnurmath and S. Chickerur, "GPU Shader Analysis and Power Optimization Model," *Engineering, Technology & Applied Science Research*, vol. 14, no. 1, pp. 12925–12930, Feb. 2024, https://doi.org/10.48084/etasr.6695.