ENF-S: An Evolutionary-Neuro-Fuzzy Multi-Objective Task Scheduler for Heterogeneous Multi-Core Processors

Athena Abdi and Armin Salimi-Badr, Senior Member, IEEE

Abstract—In this paper, an evolutionary-neuro-fuzzy-based task scheduling approach (ENF-S) to jointly optimize the main critical parameters of heterogeneous multi-core systems is proposed. This approach has two phases: first, the fuzzy neural network (FNN) is trained using a non-dominated sorting genetic algorithm (NSGA-II), considering the critical parameters of heterogeneous multi-core systems on a training data set consisting of different application graphs. These critical parameters are execution time, temperature, failure rate, and power consumption. The output of the trained FNN determines the criticality degree for various processing cores based on the system’s current state. Next, the trained FNN is employed as an online scheduler to jointly optimize the critical objectives of multi-core systems at runtime. Due to the uncertainty in sensor measurements and the difference between computational models and reality, applying the fuzzy neural network is advantageous. The efficiency of ENF-S is investigated in various aspects including its joint optimization capability, appropriateness of generated fuzzy rules, comparison with related research, and its overhead analysis through several experiments on real-world application structures of the system is determined. Otherwise, online scheduling is an efficient system-level approach. This process is performed statically at design time or dynamically at runtime. Static approaches perform exact exploration and are the appropriate choice for well-known applications where the execution structure of the system is determined. Otherwise, online schemes are required to adjust the optimization process based on the system conditions during its execution. Dynamic approaches enforce performance overhead to the system, and their optimizers are more efficient in providing a high-performance platform along with controlling other challenges such as power consumption and temperature. Consequently, they are known as the de facto standard in designing embedded and cyber-physical systems that are the most applied computing systems in real-life usages [1], [4].

There are some challenges in designing multi-core systems including performance, power consumption, chip temperature, and lifetime reliability [4], [5]. Since these systems are widely employed in autonomous applications, their power budget is limited and generally dependent on batteries. Moreover, technology advances and shrink in size of transistors increase the system’s temperature that causes hot spots and unavoidable physical damages on the chip [1], [6], [7]. In addition, these events affect the lifetime reliability of the system and make them more vulnerable to permanent faults. Since modern computing systems are widely used in safety-critical applications to improve their precision and efficiency, decreasing the lifetime reliability is another critical parameter of multi-core systems that should be considered during their design [8], [9], [10].

Dealing with the mentioned critical parameters is possible at various levels of abstraction [4], [11]. System-level methods are very appropriate due to their flexibility, low overhead, and efficiency. Task scheduling is a system-level process that assigns the application tasks to various processing cores and orders their execution regarding the system constraints. Thus, optimizing the critical parameters of multi-core systems during task scheduling is an efficient system-level approach. This process is too complicated and known as an NP-hard problem [1], [2]. Consequently, various heuristic and meta-heuristic approaches are proposed to optimize some of the mentioned criteria during task scheduling [4], [12]. Generally, task scheduling could be performed statically at design time or dynamically at runtime. Static approaches perform exact exploration and are the appropriate choice for well-known applications where the execution structure of the system is determined. Otherwise, online schemes are required to adjust the optimization process based on the system conditions during its execution. Dynamic approaches enforce performance overhead to the system, and their

I. INTRODUCTION

A LONG with technology advances, computer systems are extensively used in various aspects of life such as industry, medicine, entertainment, financial structures, communications, and so on [1], [2]. Generally, these applications have limitations in execution time that leads to high-performance computing. However, due to the power wall improving performance by increasing the frequency of processors is not efficient anymore. Thus, parallelism through integrating multiple processing cores on a chip is employed. In this context, multi-core processors and multiprocessor systems on chip (MPSoCs) provide a high level of parallelism and are the appropriate choice in designing modern computing systems. The integrated processing cores could be similar (homogeneous) or from different types (heterogeneous) [3]. Due to the different capabilities of processing cores in heterogeneous multi-core systems, they are more efficient in providing a high-performance platform along with controlling other challenges such as power consumption and temperature. Consequently, they are known as the de facto standard in designing embedded and cyber-physical systems that are the most applied computing systems in real-life usages [1], [4].
computation should be limited [13], [14]. In modern applications, the complexity of devices and their integration on a single chip makes the static estimations applied in offline approaches more uncertain. Thus, employing online control mechanisms along with static computations is essential for task scheduling [8], [14], [15].

Fuzzy inference systems (FIS) are effective solutions to properly deal with uncertainty. Since these systems are built upon fuzzy logic, they can model uncertainty mathematically in the input variables considering them as linguistic variables [16]. Moreover, online scheduling suffers from uncertainty due to employing noisy sensor measurements. Consequently, FIS is an appropriate choice for online scheduling as an uncertainty-aware controller. Fuzzy inference systems are defined in form of interpretable “IF-THEN” rules. These rules could be defined by the knowledge of experts or through a learning process. Fuzzy neural networks (FNNs) are universal approximators that combine the interpretability of FIS with the learning capability of neural networks [16], [17].

Our proposed evolutionary-neuro-fuzzy task scheduling approach (ENF-S) aims at jointly optimizing the critical parameters of multi-core systems. It employs a fuzzy neural network that is trained by a multi-objective evolutionary algorithm (NSGA-II) to optimize system execution time, chip temperature, power consumption, and lifetime reliability. During this learning phase, the rule-set of the FNN is extracted based on exploring the various design solutions to optimize the target critical parameters for several synthetic and real-life applications. These rules determine the appropriate mapping between ready tasks and the processing cores based on the system characteristics and optimizing the target critical parameters. Since the heterogeneous processing cores have various capabilities and different levels of voltage and frequencies, this mapping includes dynamic and voltage frequency scaling (DVFS) is essence.

Our proposed evolutionary-neuro-fuzzy task scheduling approach (ENF-S) aims at jointly optimizing the critical parameters of multi-core systems. It employs a fuzzy neural network trained by a multi-objective evolutionary algorithm (NSGA-II) optimizing the system execution time, chip temperature, power consumption, and lifetime reliability. During this learning phase, the rule set of the FNN is extracted based on exploring the various design solutions to optimize the target critical parameters for several synthetic and real-life applications. These rules determine the appropriate mapping between ready tasks and the processing cores based on the system characteristics and optimizing the critical parameters. Since the heterogeneous processing cores have various capabilities and different levels of voltage and frequencies, this mapping includes dynamic and voltage frequency scaling (DVFS). The main contributions of ENF-S are summarized as follows:

1) Presenting a two-phased task scheduling approach for heterogeneous multi-core systems to jointly optimize execution time, temperature, lifetime reliability, and power consumption;
2) Presenting a fuzzy neural network (FNN) to determine the criticality degree of processing cores in task assignment based on the system’s current state;
3) Presenting a learning scheme for extracting proper rules for the proposed FNN based on NSGA-II to jointly optimize the main critical parameters of heterogeneous multi-core systems;
4) Presenting a lightweight and application-independent online scheduler based on the learned FNN;
5) Utilizing the improvement techniques including DVFS and adding cooling slacks to explore the design space and perform optimization efficiently.

The rest of this article is as follows: First previous studies are reviewed in Section II. Afterward, the necessary preliminaries are presented in Section III. Section IV explains our proposed method (ENF-S). Experimental results to evaluate the efficiency of ENF-S and compare it with the related studies are presented in Section V. Finally, the conclusions and future trends are presented in Section VI.

II. RELATED WORK

To deal with optimizing the critical parameters of multi-core processors (including power consumption, reliability, and performance) during their scheduling, an NP-hard constrained multi-objective problem forms. Thus, several heuristic, meta-heuristic and limited exact approaches tried to address it [1], [4]. Optimizing these critical parameters is often considered partially in related research, with concentrate on system performance and energy consumption.

Heuristic approaches try to optimize the critical parameters through an efficient and lightweight approach [18]. Zhou et al. proposed three heuristic approaches based on consideration of various critical parameters of multi-core systems [19]. In these methods, energy, execution time, and lifetime reliability are considered as objectives and constraints of optimization problems, interchangeably to construct a single-objective optimization problem. Sheikh and Ishfaq presented heuristic scheduling approaches to optimize the performance, power consumption, and temperature jointly [20]. These methods mainly integrate the objectives by a weighted sum that leads to one optimal point in the multi-objective optimization process. These weights could be various in different applications.

Salami et al. proposed a fairness and energy-aware task scheduler for heterogeneous multi-core processors [21]. In this approach, the cores classify into various clusters based on their capabilities and limitations. Then the programs are ranked based on their execution and energy requirements. By determining the best mapping among all processors and programs, the energy consumption and fairness are optimized through frequency scaling in this research. To determine the ideal hardware for each program, da Silva et al. proposed a statistical regression-based method [22]. This predictor is implemented on a compilation strategy and can optimize execution time and energy consumption during the mapping process. Moreover, Bhat et al. presented a prediction-based scheduling approach to consider the energy, temperature, and performance of MP-SoCs, simultaneously [23]. This method adapts the predicted non-ideal power and temperature by exploring the design space by gradient search algorithm (GSA) and selects the appropriate
mapping decisions that optimize these parameters along with performance. Ehttari et al. proposed a task scheduler based on a fuzzy inference system to optimize power consumption and temperature simultaneously [24]. In this approach, the static and predefined fuzzy rules are employed to determine the appropriate processing core and the scheduling details are defined based on improving the system temperature and utilization. These approaches are static and applied to the system based on a design time computation that arises from the static behavior of the system.

Meta-heuristic approaches try to explore the solution space and optimize the critical parameters during the task scheduling process more efficiently [4]. The genetic algorithm is one of the most effective meta-heuristic algorithms and is compatible with task scheduling due to its operators [25]. Abdi and Zarandi proposed a static task scheduling by an NSGA-II-based engine to optimize performance, lifetime reliability, power consumption, and temperature of heterogeneous MPSoC [26]. Since these approaches explore the design space more comprehensively than heuristic methods, their precision is at the cost of high processing time. Moreover, there exist exact methods that model the problem as integer linear programming (ILP) for small-sized applications to have a general solution and evaluate the efficiency of heuristic and meta-heuristic methods [27]. Some studies proposed the ILP-based scheduling models along with simple heuristic approaches for optimizing various critical parameters during the task scheduling [5], [28]. Making these exact models more comprehensive and scalable to improve their efficiency and precise inexact solutions is studied in [27].

Today due to the various applications of MPSoCs and their dynamic behaviors, considering static scheduling approaches based on a predefined scenario, the same as most mentioned related studies, is not enough. Employing hybrid or dynamic schedulers that are adaptive to the system application and its state during execution is required [4], [14]. Since the complexity of the online control schemes affects the system performance directly, limited lightweight approaches such as threshold checking are mainly applied [11], [13], [14]. Besides, a single characteristic of tasks like their deadline or repetition rate considered as real-time parameters, such as failure rates, and it leads to their uncontrolled adjustments [8], [14]. Thus, applying a proper online controller to decide based on various ranges of the critical parameters along with managing them simultaneously during the system execution is crucial. To this aim, fuzzy neural networks that combine the interpretability, fuzziness and learning capability could be a proper choice in performing the online scheduling process. Thus, our proposed ENF-S deals with this problem by employing a light weight fuzzy neural network-based scheduler in heterogeneous multi-core systems.

III. PRELIMINARIES

A. Application and Architecture Models

We consider the application as a directed acyclic graph (DAG) consists of tasks as nodes and their data dependency as edges [1], [4], [29]. This application graph (\(G_{\text{App}}\)), is formally defined in 5-tuple: \((V_{\text{App}}, E_{\text{App}}, W_{\text{App}}, D_{\text{App}}, WCET_{\text{App}})\). Where these tuples represent the tasks \((V_{\text{App}})\), their data dependency relation \((E_{\text{App}})\), cost of communication \((W_{\text{App}})\), application deadline \((D_{\text{App}})\) and worst-case execution time of each node \((WCET_{\text{App}})\).

The data dependencies of the tasks are defined as the predecessor-successor relation. This communication has a cost represented as the weight of the edges in \(G_{\text{App}}\). Moreover, due to the intensive employment of embedded systems in real-time applications, they should finish their mission before a predefined deadline. This deadline is considered for each application and defines as the time by which the application should finish its execution. Furthermore, the worst-case execution time of each task of the application is determined based on the lowest operational frequency level of the target processor (nominal frequency). This parameter shows the most pessimistic execution time of each task on the processing platform. It is possible to have multiple entries and exit tasks in each application graph, and the task execution is considered a non-preemptive process. Fig. 1(a) shows an example of the defined application graph \((G_{\text{App}})\) for the Office application of E3S benchmark suite [30].

Our considered architecture is a heterogeneous multi-core system modeled as a graph \((G_{\text{Arc}})\). In \(G_{\text{Arc}}\), the nodes represent the heterogeneous processing cores, and the edges show their communication links. The heterogeneity of processing cores is defined in their execution capability inspired by the ARM’s big.LITTLE architecture that integrates high-performance and low-power clusters on a chip. Each processing core has multiple levels of operational voltage/frequency during the execution [21], [31]. Fig. 1(b) shows a sample architecture graph \((G_{\text{Arc}})\) of a heterogeneous quad-core platform inspired by the ARM’s big.LITTLE structure.

B. Critical Parameters of Multi-Core Systems

The most important critical parameters of multi-core processors are performance, power consumption, reliability, chip temperature, cost, and area. There are antagonistic relations among these parameters that make their joint optimization during design more complicated [4], [5], [23]. The system-level models of these parameters are presented in the rest of this section.
1) Lifetime Reliability: Multiprocessor systems are vulnerable to transient and permanent faults occurring in processing units and commutations. Transient faults have temporal effects on the system, while permanent ones affect the system’s lifetime. The lifetime reliability of the multiprocessor system is threatened by several failure mechanisms with intrinsic sources. These failure mechanisms are summarized in three classes:

- Electro-migration (EM) and Stress-migration (SM) on the communication links,
- Time-dependent dielectric breakdown (TDDB) in gate oxide,
- Negative Bias Temperature Instability (NBTI) in transistors.

Electro-migration and Stress migration occur due to undesirable atom movements in the interconnections. They depend on the current density and thermal incompatibility of metal and dielectric, respectively. The empirical mean time to failure (MTTF) relations of these failure mechanisms are derived as follows [32], [33]:

\[
MTTF_{EM} = A_{EM} \times (J - J_{circ})^{(-n)} \times \exp \left( \frac{E_a}{KT} \right) \tag{1}
\]

\[
MTTF_{SM} = A_{SM} \times (T_0 - T)^{(-n)} \times \exp \left( \frac{E_a}{KT} \right) \tag{2}
\]

where, \( A_{EM} \) and \( A_{SM} \) are the scale factors, \( J \) and \( J_{circ} \) are the applied and threshold current densities, \( n \) is the current density exponent, \( T_0 \) is stress-free temperature for metal, \( E_a \) is the activation energy, \( K \) is Boltzmann’s constant, and \( T \) represents the temperature in Kelvins.

Time dependent dielectric breakdown (TDDB) occurs due to the gradual degradation of the gate dielectric. The MTTF relation of this mechanism is computed as follows [32], [33]:

\[
MTTF_{TDDB} = A_{TDDB} \times \exp(-\gamma E_{ox}) \times \exp \left( \frac{E_a}{KT} \right) \tag{3}
\]

where, \( A_{TDDB} \) is a scale factor, \( \gamma \) is field acceleration, and \( E_{ox} \) is the electric field across the dielectric.

Negative bias temperature instability (NBTI) leads to increasing the threshold voltage of transistors. Changes in threshold voltage causes timing violations and failure of the system. The MTTF relation of this mechanism is computed as [32], [33]:

\[
MTTF_{NBTI} = A_{NBTI} \times (V_{GS})^{-\gamma} \times \exp \left( \frac{E_a}{KT} \right) \tag{4}
\]

where, \( A_{NBTI} \) is a scale factor, \( V_{GS} \) is the absolute value of the gate voltage, and \( \gamma \) is the voltage acceleration factor. The presented models of failure rates are experimentally derived and standardized by JEDEC and updated based on the current technology [9], [14], [33], [34].

In our proposed ENF-S, to consider the effect of the explained failure mechanisms jointly, we integrate them by sum-of-failure-rates (SOFR) measure [9]. Thus, the system’s MTTF based on SOFR measure is computed as follows:

\[
MTTF = \frac{1}{\lambda_{system}} = \frac{1}{\sum_{p=1}^{i} \sum_{k=1}^{l} \lambda_{pl}} \tag{5}
\]

where \( \lambda_{pl} \) represents the failure rate of \( p^{th} \) processing unit because of the \( l^{th} \) failure mechanism, and \( \lambda_{system} \) presents the described SOFR measure.

Since the approximated failure rate is related to time, we utilize the GSFR (global system failure rate) measure to consider the failure rate and execution time in an unified relation [5], [35]. The GSFR of a schedule that consists of application tasks is computed as follows:

\[
\Lambda = \frac{\lambda_{system}}{\sum_{(\tau_i, c_i, f_i) \in S} T x \exp \left( \tau_i, c_i, f_i \right)} \tag{6}
\]

here, \( \lambda_{systems} \) is the failure rate of scheduling based on the described SOFR, and \( T x \exp \) is the total execution time of the application’s tasks on the assigned processing cores and frequency levels.

2) Power Consumption: Multi-core systems are widely used in autonomous embedded applications which are battery-dependent and have limitation in energy resources. Thus, managing power consumption during their design is required. Power consumption consists of dynamic and static aspects. Dynamic power is dependent on the operational voltage and frequency, while the static part is mainly related to the leakage power and system temperature. Thus, the overall power consumption of processors is derived as follows:

\[
P_{system} = P_{dynamic} + P_{static} = C_{eff} \times V^2 \times f + \alpha \times T(t) + \beta; \tag{7}
\]

where \( C_{eff} \) is the switching capacitance, \( V \) and \( f \) are operational voltage and frequency, \( \alpha \) and \( \beta \) are architecture-dependent coefficients [7], and \( T \) represents the chip temperature in Kelvin.

3) Chip Temperature: Along with technology advances, increase in power density and rate of permanent failures, the chip temperature has become important. The temperature of processors based on the RC equivalent thermal model is computed as follows [7], [34]:

\[
C \left( \frac{dT(t)}{dt} \right) + G (T(t) - T_{amb}) = P(t); \tag{8}
\]

where \( C \) and \( G \) are thermal capacitance and conductance, \( t \) is time, \( T(t) \) is instantaneous and \( T_{amb} \) is the ambient temperature. \( P \) expresses the power consumption of the system that is described in (7).

Moreover, the temperature of each core of a multi-core system is affected by its neighbors’ heat. To consider this, here we use the following two-dimension spatial heat transfer equation among the processing cores [34]:

\[
Heat transfer_{2D} = \sum_{c \in nbr(c)} G(c, c') (T_c(t) - T_{c'}(t)) \tag{9}
\]

where \( nbr(c) \) is the neighbor set of core \( c \), \( T_c \) and \( T_{c'} \) are the temperatures of each core and its neighbor \( c' \), \( G(c, c') \) is the thermal conductance between cores \( c \) and \( c' \).

By applying the power consumption as (7), heat transfer between adjacent cores as (9), and solving the differential equation of (8), the temperature of each processing core can be
expressed as:

\[ T_v(t) = T\infty + (T(t_0) - T\infty) e^{-a(t-t_0)}, \quad \text{where} \]

\[ T\infty = \frac{b}{a} \]

\[ a = \frac{G - \alpha + \sum_{c'\in\text{nbr}(c)} G(c, c')}{C} \]

\[ b = \frac{G \cdot T_{\text{amb}} + \sum_{c'\in\text{nbr}(c)} T_v(t) \cdot G(c, c') + C_{\text{eff}} \cdot V^2 \cdot f + \beta}{C} \]

(10)

where \( T(t_0) \) and \( T\infty \) are the initial and steady state temperatures, respectively [7], [34]

C. Multi-Objective Optimization Methods

Multi-objective optimization optimizes two or more conflicted criteria during a decision-making problem. Due to the existing trade-off among objectives, the final solution is not unique. For instance, minimizing power consumption and maximizing the performance of a chip are not aligned, and greedy selection of the best performance leads to losing power consumption.

Pareto optimization is an appropriate approach considering the mentioned challenges. It explores the whole solution space and utilizes the dominance rule to compare the generated solutions and meet their existing trade-off. The set of dominance points in a solution space constructs the "Pareto Front." Constructing the whole Pareto front and simultaneous consideration of all objectives are not trivial. To this aim, various heuristic schemes such as aggregating, hierarchization, transformation, and population-based methods are utilized [5], [35]. The three former approaches convert the problem to a single objective form that reduces accuracy. However, the later ones like NSGA-II explore the whole design space and build the Pareto front more comprehensively [36].

IV. PROPOSED METHOD

Our proposed task scheduling approach (ENF-S) aims at optimizing the critical parameters of heterogeneous multi-core systems including execution time, temperature, lifetime reliability, and power consumption. Since task scheduling is dependent on the current state of the system, we model it as a function assigning ready tasks to the proper processing cores based on analyzing the system’s critical parameters. To this aim, we have employed an online scheduler using a fuzzy neural network (FNN) as a lightweight function approximator. To learn this FNN, we have applied NSGA-II on several real-world and synthetic application graphs. The details of ENF-S are explained in this section.

A. Problem Statement

The assumptions and objectives of our proposed ENF-S could be formulated as follows:

\textbf{Given:}

\begin{itemize}
  \item The architecture of the target multi-core systems in terms of its processing cores and their interconnections (\( G_{Ar,c} \)),
  \item The set of real-life and synthetic applications modeled as DAG of tasks and their data dependencies (\( G_{App} \)) to train and test our proposed scheduler,
  \item The set of supported operational voltage and frequency levels of each processing core (\( C \)) as their heterogeneity specifications (\( C_{\text{eff}} \)),
  \item The matrix of worst-case execution time (WCET) of each task (\( \tau \)) on the various and heterogeneous processing cores (\( WCET[\tau, C] \)),
  \item The valid range of lifetime reliability (\( \Lambda \)), power consumption (\( P \)) and temperature (\( \theta \)) of architecture type for constructing fuzzy rules.
\end{itemize}

\textbf{Goals:}

\begin{itemize}
  \item Learning the proposed fuzzy neural network-based scheduler using NSGA-II such that:
    \begin{itemize}
      \item Optimizing the critical parameters of heterogeneous multi-core systems by appropriate mapping of tasks to processing cores and setting their voltage and frequency levels;
      \item Distributing tasks on heterogeneous processing cores to balance their wear-out rate;
      \item Determining a proper set of interpretable fuzzy rules to meet the existing trade-offs among the defined objectives.
    \end{itemize}
  \item Online scheduling and mapping of application tasks on heterogeneous processing cores through the learned fuzzy neural network scheduler such that:
    \begin{itemize}
      \item Mitigating the existing gap between static models and sensors’ data during execution,
      \item Making online decisions based on the extracted fuzzy rules to apply runtime modifications based on the state of the system.
    \end{itemize}
\end{itemize}

B. Scheduling Algorithm

We propose an online task scheduler (ENF-S) for heterogeneous multi-core systems to jointly optimize \textit{execution time, chip temperature, lifetime reliability, and power consumption} using a learned fuzzy neural network. First, it is required to determine the candidate tasks at each scheduling stage due to the existing precedence relations among the application’s tasks. These candidate tasks form a ready list. Then based on the available processing cores, a specific number of ready tasks are selected and executed in parallel. To determine the execution priority of ready tasks, we define an \textit{emergency} criterion considering the application deadline and the worst-case execution time of each task. This criterion demonstrates the flexibility of each task and its delay tolerance during the scheduling. Thus, the defined \textit{emergency} criterion derives at each scheduling step as follows:

\[ \text{urgent_task}(n) = \arg \min_{\tau \in \text{Ready}(n)} (D_{\text{App}} - WCET_{\tau}) \]  

(11)

where, \( n \) represents the scheduling step, \( \tau \) is the ready task, \( D_{\text{App}} \) is the application deadline, and \( WCET_{\tau} \) is the worst-case execution time of the ready task \( \tau \) at nominal frequency. The tasks in ready list are sorted based on \textit{emergency} criterion in descending order to prioritize the most urgent tasks during scheduling process.
Afterward, it is required to determine the appropriate processing core with the operational voltage and frequency level for executing each selected task. Thus, based on the available processing cores and considering the critical parameters, a specific number of ready tasks are assigned. Selecting the best execution option based on the available processing cores and voltage/frequency levels is a multi-objective optimization problem. To solve this problem, ENF-S employs a fuzzy neural network. In this context, we introduce a criticality degree for each processing core of the multi-core system based on its current load, temperature, power consumption, and wear-out status. This criticality degree is calculated by our learned fuzzy neural network. To enable DVFS each processing core supports multiple operational voltage and frequency levels in execution. Therefore, it is also required to determine the appropriate frequency levels of executing each task. The selected frequency is affected by existing trade-offs among the critical parameters. For instance, higher frequencies improve the execution time in cost of losing power, temperature, and lifetime reliability.

Our proposed neuro-fuzzy scheduler (ENF-S) is trained by NSGA-II, as one of the most effective multi-objective algorithms to jointly optimize the critical parameters of multi-core systems at design time. Fuzzy neural networks are interpretable universal approximators. Thus, by properly defining their rules, they can schedule various application graphs at runtime. To this aim, we have applied the learning process using NSGA-II at design time for several application graphs. In this way, the thorough exploration capability of the population-based methods is embedded in the defined fuzzy rules of ENF-S and passed to runtime as a lightweight and general fuzzy scheduler.

At each scheduling step, the ready list of tasks, the available processing cores, and their operational voltage and frequency levels are given as inputs to the proposed fuzzy neural network-based scheduler. This scheduler calculates a criticality degree for each processing core in terms of its current load, power consumption, temperature, and wear-out for executing each ready task. Thus, the mapping decision with the lowest criticality degree is the best to jointly optimize execution time, chip temperature, lifetime reliability, and power consumption of the target multi-processor system. The described process is repeated iteratively for various ready tasks during the task scheduling process. Along with task scheduling, the values of the main parameters of the underlying architecture are updated and compared to the system limitations.

The explained process repeats for several training application graphs to learn the appropriate values of the fuzzy neural network’s parameters through NSGA-II. The details of our proposed task scheduling approach are summarized in Algorithm 1. Moreover, the details of our utilized learning approach and fuzzy neural network architecture are explained in Sections IV-D and IV-C.

### C. Fuzzy Neural Network

In the proposed method, a fuzzy neural network (FNN) is presented to determine the criticality degree of each core based on its current state. Fig. 2 shows the detailed architecture of the proposed FNN. This FNN consists of five layers: 1- input layer, 2- fuzzification layer, 3- fuzzy rules layer, 4- normalization layer, and 5- output layer. The description of each layer is outlined as follows:

1- **Input Layer.** Regarding the target objectives, this layer consists of four neurons that deliver the variables that describe the state of each core along with its associated voltage/frequency levels as $x_1$ = execution time ($ex(e)$), $x_2$ = temperature ($\theta$), $x_3$ = failure rate ($GSFR$), and $x_4$ = power consumption ($P$).

2- **Fuzzification Layer.** Each neuron of this layer receives an input variable from the input layer and calculates its membership degree to a lingual value based on a triangular membership function. The membership value of the $j^{th}$ input variable $x_j$ (for $j = 1, 2, 3, 4$) to the $l^{th}$ fuzzy set $A_l$ is calculated as follows:

$$
\mu_{A_l}(x_j) = \begin{cases} 
\frac{(x_j - c_l)}{(b_l - a_l)} & a_l \leq x_j \leq c_l \\
\frac{(b_l - x_j)}{(b_l - c_l)} & c_l \leq x_j \leq b_l \\
0 & \text{otherwise}
\end{cases}
$$

where $a_l < b_l < c_l$ are start value, center value, and end value of $l^{th}$ triangular membership function.

![Fig. 2. The detailed architecture of FNN. The network receives the current state of a processing core regarding the selected ready task as the input (execution time, temperature, failure rate, power consumption), and calculates its criticality degree.](image)

**Algorithm 1: Proposed Scheduling Method (ENF-S).**

```plaintext
1. n ← 1
2. ready_list ← tasks of $G_{App}$ without data dependencies
3. while ready_list ≠ ∅ do
4. $\tau_{urgent} = \text{argmax}_{\tau_{ready list}} (D_{App} - WCET_{c})$
5. for $(c \in \text{cores and } f \in \text{v/f levels})$ do
6. $\text{exe} \leftarrow \text{exe}(\tau_{urgent}, c, f)$
7. $\Lambda \leftarrow \Lambda(\tau_{urgent}, c, f)$
8. $P \leftarrow P(\tau_{urgent}, c, f)$
9. $\theta \leftarrow \theta(\tau_{urgent}, c, f)$
10. $\text{Criticality Degree}(c,f) \leftarrow \text{FNN} (\text{exe}, \Lambda, P, \theta)$
11. end
12. $\text{Selected_core} \leftarrow \text{argmin}_f \text{Criticality Degree}(c,f)$
13. Schedule[n] ← $(\tau_{urgent}, \text{Selected_core})$
14. end
15. n ← n+1
```
Algorithm 2: FNN for Task Scheduling.

1 - Initialization:
2 \( n^f \): number of objectives, \( n_{f_s} \): number of fuzzy sets, \( a,b,c \):
3 \{fuzzy sets’ parameters\}/
4 \( \{n, n_{f_s} \} \leftarrow \{4,5\} \)
5 \( \{a, b, c\} \leftarrow \{[0 : 0.25 : 0.75], [0 : 0.25 : 1], [0.25 : 0.25 : 1]\} \)
6 - Input Layer:
7 \( x_1 \leftarrow exe, x_2 \leftarrow \theta, x_3 \leftarrow \Lambda, x_4 \leftarrow P \)
8 - Fuzzification Layer:
9 for \( j=1:n \) do
10 for \( i=1:n_{f_s} \) do
11 compute \( \mu_{A_i}(x_j) \) (eq. (12))
12 end
13 end
14 \( R \leftarrow n_{f_s} \), \( \triangleright \) number of fuzzy rules*/
15 for \( i=1:R \) do
16 compute strength of the \( i^{th} \) rule \( (f_i) \) (eq. (13))
17 end
18 - Normalization Layer:
19 for \( i=1:R \) do
20 compute the normalized strength of the \( i^{th} \) rule \( (\phi_i) \)
\( \triangleright \) (eq. (14))
21 end
22 - Output Layer:
23 compute the final criticality degree \( y \) (eq. (15))

3- Fuzzy Rules Layer. The neurons of this layer apply a T-Norm operator on outputs of the previous layer to calculate rules’ firing strength value. In the proposed method, the minimum function is chosen as the T-Norm operator. Therefore, the output of \( i^{th} \) neuron as the firing strength of \( i^{th} \) fuzzy rule is calculated as follows:

\[
 f_i = \min_j (\mu_{A_j}(x_j)) \tag{13}
\]

where \( \mu_{A_j}(x_j) \) indicates the membership value of \( j^{th} \) input variable to the fuzzy set involved in \( i^{th} \) rule related to the \( j^{th} \) dimension \( (A_j) \), computed as an output value of the previous layer’s neurons.

4- Normalization Layer. The neurons of this layer calculate the rules’ normalized firing strength as follows:

\[
 \phi_i = \frac{f_i}{\sum_{i=1}^{R} f_i} \tag{14}
\]

where \( R \) is the number of fuzzy rules.

5- Output Layer. Finally, the output layer’s neuron calculates the defuzzified final criticality degree which would be utilized in scheduling algorithm (see Section IV-B) as follows:

\[
 y = \sum_{i=1}^{R} \phi_i y_i \tag{15}
\]

where \( y_i \) is the proposed output of the \( i^{th} \) rule. The details of FNN’s function is summarized in Algorithm 2.

D. Learning Method

Regarding the controlling role of the proposed fuzzy neural network (FNN), we consider a potential rule for each permutation of different lingual values of input variables. In the scheduling problem, the range of each input variable is predetermined. Therefore, we first normalize each input variable to the range of \([0,1]\). Next, we partition the range of each input variable uniformly. Fig. 3 shows the uniformly partitioning of input variable \( x \).

Uniformly partitioning of the input variables based on Fig. 3 provides the values of the premise parts’ parameters \( (a_i, b_i, \text{and } c_i \ (i=1,2,...,R) \text{ in } (12)) \). By considering five fuzzy sets ("Very Low (VL)", "Low (L)", "Medium (M)", "High (H)", and "Very High (VH)" like Fig. 3), a,b, and c of the formed triangular fuzzy sets would be \( a = [0, 0.25, 0.5, 0.75], b = [0, 0.25, 0.5, 0.75, 1], \text{ and } c = [0.25, 0.5, 0.75, 1] \). The consequent part’s parameter of each fuzzy rule \( (y_i \text{ in } (15)) \) is the criticality degree of each core assigning the selected ready task based on \( i^{th} \) rule. To determine these values we employ a multi-objective evolutionary algorithm (NSGA-II) considering all important scheduling criteria through a learning scheme.

NSGA-II is a population-based evolutionary algorithm that considers several objectives in sorting and selecting the individuals of the next generation. This algorithm provides non-dominant solutions by forming a Pareto front. To use the NSGA-II, the fitness of each individual is defined based on minimizing the 4-tuple vector cost function presented as follows:

\[
 Cost = \left[ (exe(C), \theta(C), \Lambda(C), P(C)) \right] \tag{16}
\]

where, \( exe(C), \theta(C), \Lambda(C), \text{ and } P(C) \) are execution time, temperature, failure rate (GSFR), and power consumption of core \( C \). To learn the parameters of the consequent part each individual is considered as a solution for scheduling as an FNN. Since the premise parts’ parameters are determined based on the uniform partitioning (see Fig. 3), to encode the FNN as an individual of the evolutionary algorithm, only the consequent parts’ parameters are considered as the genes of each chromosome. To calculate fitness values, the tasks of each application graph is executed based on the proposed scheduling of each individual and the mentioned critical parameters are measured as its fitness.

The NSGA-II final result would be a Pareto front that consists of a set of non-dominant optimal solutions. Each member of the Pareto front is an FNN that considers various objectives with different concentrations. Here, the Pareto front consists of solutions that are extremely focused on one objective or the ones that jointly consider various objectives. It should be noted that all these solutions meet the existing constraints on various objectives defined in the system. To select one of these solutions as our FNN, we choose the point with the minimum distance...
Algorithm 3: Learning Algorithm.

1. \( y \leftarrow \text{zeros}(R) \)
2. \( T \leftarrow \text{Training application graphs} \)
3. for \( t = 1 : \text{length}(T) \)
4. \( \text{Extracting Pareto Front of } G_{\text{App}}[t] \text{ by NSGA-II} \)
5. Choose the middle point of Pareto Front \( (y_{\text{mid}}) \)
6. \( \text{MSD} \leftarrow \text{zeros(length(ParetoFront))} \)
7. for \( p_i, p_j \in \text{Pareto Front} \)
8. \( \text{MSD}[i] \leftarrow \text{MSD}[i] + \frac{\text{length}(\text{ParetoFront})}{||p_i - p_j||} \)
9. end
10. \( y_{\text{mid}} \leftarrow \arg\min_{\text{MSD}}(\text{MSD}) \)
11. \( y_{\text{out}} \leftarrow y_{\text{out}} + y_{\text{mid}} \)
12. end
13. \( y_{\text{out}} \leftarrow y_{\text{out}}/\text{length}(T) \)

from the other solutions (mean-squared-distance (MSD)). In this way, we discard the extreme solutions and focus on the middle points of the Pareto front that considers all criteria moderately.

A training dataset including different application graphs with various sizes and attributes is used to learn the consequent parts’ parameters. For each benchmark, a Pareto front is extracted and a solution that has the minimum mean-squared-distance (MSD) is chosen from all the extracted solutions. Next, the final solution is calculated by averaging the chosen solutions of different problems. The learning algorithm is summarized in Algorithm 3. During the scheduling process, it is possible that the premise parts of some rules (mainly the extreme values) are not satisfied. Indeed all rules of a solution do not fire through solving a problem. Therefore we prune the inactivated rules (rules that their firing rate is lower than a predefined threshold) from the final fuzzy rules.

V. EXPERIMENTAL RESULTS

In this section the effectiveness of our proposed method is demonstrated by several experiments.

A. Simulation Setup

Our proposed scheduling approach is implemented on a simulated heterogeneous MPSoC platform inspired by Arm big.LITTLE architecture. This system consists of four processing cores: two high performance processors of ARM Cortex-A15 and two low power processors of ARM Cortex-A7. Each processor is equipped with three operational voltage and frequency levels to employ DVFS property. The details of the considered system architecture are presented in Table I.

![Algorithm 3: Learning Algorithm.](image)

Our simulation platform is constructed by integrating precise and well-known system simulators to model the architecture of heterogeneous MPSoC as well as possible. To this aim, we employ GEM5 full-system simulator [37] to simulate the underlying platform. Moreover, McPAT framework [38] is utilized to model the power consumption of the system. Furthermore, we employ QUILT [39] along with Hotspot [40] to build the target floorplan of the system and model its temperature, accurately. The appropriateness of these tools and their integration in modeling big.LITTLE ARM architecture over real systems such as Samsung Exynos 5 Octa and ODROID-XU3 are studied and validated [31], [41].

Our simulation platform uses MiBench [42] and PARSEC [43] benchmark suits to generate the power and thermal traces of the system. Moreover, some parts of MiBench suit are then employed during the learning process. This information leads to our system-level simulator to execute various scenarios based on our proposed evolutionary-neuro-fuzzy task scheduling approach. Fig. 4 presents our considered simulation process and the integration flow of employed tools. The simulation flow is mainly inspired by previous studies [15], [41], [44].

Furthermore, our online scheduler employs temperature sensors using the Hotspot tool that provides an accurate RC thermal model based on our assumed system architecture [40]. The accuracy of this tool is exactly studied and its performance is very close to real thermal sensors [8].

B. Datasets and Learning Setup

The learning process aims at determining the proper fuzzy rules for online scheduling considering the critical parameters. This process is implemented using an NSGA-II-based optimization engine. In this context, two separated datasets including real-world and synthetic application graphs are considered the training and test datasets. To cover various applications and scenarios, task graphs with different sizes and types are considered in both training and test datasets. Application graphs of sizes from 5 to 85 nodes are selected from E3S, based on data from the Embedded Microprocessor Benchmark Consortium (EEMBC), MiBench, and various synthetic graphs generated by TGFF [30],
and their constant parameters are mainly shows the values of applied hyper-
To show the joint optimization capability of the proposed
L[45] to (d).
and are presented in Table
such as deadline are
One of the advantages of our proposed ENF-S is its
T
[42]

[38x379],
[38x753]ABDI AND SALIMI-BADR: ENF-S: AN EVOLUTIONARY-NEURO-FUZZY MULTI-OBJECTIVE TASK SCHEDULER 487

adapted from
plained in Section
and lifetime reliability is implemented. These models are ex-
graphs that are mentioned in Section
all graphs of that category. The main parameters of application
table, the real-life entries (consumer, telecom, office, etc.) point
training and the rest for test. The characteristics of the training
various application graphs with different sizes have been em-
is directly dependent on its size. Since in the training of ENF-S,
be mentioned that the training process of each application graph
To consider fairness during the learning process, we
randomly selected about 70% of the mentioned graphs for
training and the rest for test. The characteristics of the training
and test datasets are summarized in Table II. To summarize this
table, the real-life entries (consumer, telecom, office, etc.) point
at a class of applications including multiple programs based on
E3S and MiBench suits, and their size is reported on average for
all graphs of that category. The main parameters of application
graphs that are mentioned in Section III such as deadline are
defined for random graphs by the TGFF tool automatically.
It should be noted that, the task graphs that are the input of
our proposed method are extracted from the executable code
of the applications. These codes are available for real-world
benchmarks and for random cases, it is possible to generate
dummy codes based on the dependencies and execution time of
the applications. These codes are available for real-world

Furthermore, to better study the satisfied trade-offs, three
points of the 2D view of the Pareto front (Fig. 5(b) to (d))
with various behaviors in terms of our considered objectives are
analyzed. Point 2, has the highest execution time but its tempera-
ture, power consumption, and lifetime reliability are the lowest.
Moreover, point 8 reflects moderate behavior due to its median
In this figure, solution points with lower temperature, power,
offs among the considered critical parameters can be deduced.

C. Simulation Results
To evaluate the efficiency of our proposed evolutionary-based
neuro-fuzzy scheduler, in solving the target problem and com-
paring it to related studies, three classes of experiments are con-
sidered and presented in this section. First, the joint-optimization
ability of the method is investigated by presenting the surface of
the extracted Pareto front. Next, the interpretability of the model
is studied by presenting the extracted fuzzy rules. Afterward,
the performance of the method is compared with some related
approaches.

1) Studying Joint Optimization Capability: Description and
Aims. To show the joint optimization capability of the proposed
method, its final Pareto front in various views based on a sample
mid-sized graph of size 40 is shown in Fig. 5. In Fig. 5(a), all
target objectives are considered at the same time. To show the
four-dimensional Pareto front in this figure, three axes are set to
power, temperature, and lifetime reliability while the colors of
different solution points reflect their execution time. Moreover,
the 4D Pareto front (Fig. 5(a)) is expanded by projection of each
generated solution point into two dimensions in Fig. 5(b) to (d).
In these figures, three different projections are considered based
on variations of execution time over temperature, power con-
sumption, and failure rate. The solutions points are enumerated
to make them traceable in various viewpoints and compare their

Results and Discussion. Based on Fig. 5(a), the existing trade-
offs among the considered critical parameters can be deduced.
In this figure, solution points with lower temperature, power,
and failure rate values have red colors that represent their higher
execution time. Reversely, blue points represent the solutions
with lower execution time but with higher costs of temperature
and power consumption. As this figure shows, the solution
points have the proper color and spatial dispersion; therefore,
our ENF-S explores the whole design space and extracts rules
with efficient space coverage.

Moreover, to learn the proposed neuro-fuzzy scheduler, the
system model including the temperature, power consumption,
and lifetime reliability is implemented. These models are ex-
plained in Section III and their constant parameters are mainly
adapted from [7], [33] and are presented in Table III. It should
be mentioned that the training process of each application graph
is directly dependent on its size. Since in the training of ENF-S,
various application graphs with different sizes have been em-
ployed, the average epoch time for each graph is estimated at
about 0.05 sec for each individual in each generation of the
NSGA-II. Finally, Table IV shows the values of applied hyper-
parameters of our proposed NSGA-II-based learning approach.

[42], [45]. To consider fairness during the learning process, we
have randomly selected about 70% of the mentioned graphs for
training and the rest for test. The characteristics of the training
and test datasets are summarized in Table II. To summarize this
table, the real-life entries (consumer, telecom, office, etc.) point
at a class of applications including multiple programs based on
E3S and MiBench suits, and their size is reported on average for
dl the 4D Pareto front (Fig. 5(a)) is expanded by projection of each
generated solution point into two dimensions in Fig. 5(b) to (d).
In these figures, three different projections are considered based
on variations of execution time over temperature, power con-
sumption, and failure rate. The solutions points are enumerated
to make them traceable in various viewpoints and compare their
corresponding values.

| TABLE II |
| --- |
| CHARACTERISTICS OF THE TRAINING AND TEST DATASETS |

| Application Graphs | Range of Size | Application Graphs | Range of Size |
| --- | --- | --- | --- |
| Gaussian elimination | 9 | Office | 5 |
| Consumer | 12 | Networking | 13 |
| Telecom | 30 | Auto-indust | 24 |
| Small-scaled Random | 8 to 20 | Small-scaled Random | 8 to 20 |
| Mid-scaled Random | 23 to 50 | Mid-scaled Random | 22 to 50 |
| Large-scaled Random | 52 to 85 | Large-scaled Random | 52 to 85 |

| TABLE III |
| --- |
| PARAMETER VALUES OF THE SYSTEM MODEL |

| Model | Parameter Value |
| --- | --- |
| Temperature | $C = 0.03 J/K, G = 0.3 W/K$; $T_{amb} = 293 K, G_{electrical} = 0.1 W/K$ |
| Lifetime | $t_d (ENF, SM) = 1044.45 s$, $J = 159$, $n_{rel} = 1.1$, $n_{rel} = 2.5$, $n_{bvt} = 5$, $t_{ddr, bvt} = 78, -0.08$ |
| Power Consumption | $\alpha = 0.1 W/K, \beta = -11 W/KV$ |

| TABLE IV |
| --- |
| HYPER-PARAMETERS’ VALUES OF THE PROPOSED LEARNING METHOD |

| Hyper-parameters | Value |
| --- | --- |
| Population Size | 200 |
| Cross-over Probability | 40% |
| Mutation Probability | 70% |
| Number of Generations (iterations) | 500 |
| Number of Fuzzy Sets | 5 per each objective |
| Number of Training Graphs | 912 |
| Number of Test Graphs | 401 |
Fig. 5. Extracted Pareto front: (a) sketched in four-dimension view (temperature ($\theta$), power ($P$), and failure rate ($\Lambda$) as axes and execution time is encoded in the colors. (b-d) sketched in two-dimension views where temperature, power consumption, and failure rate are plotted versus the execution time. Solution points are enumerated to make them traceable among various views.

Fig. 6. The most frequently fired fuzzy rules generated by ENF-S for a sample solution. Columns are the input/output variables and the rows indicate the fired rules’ indexes. The colors from blue to red indicates the lower values to the higher ones.

Since these rules utilize linguistic variables, their appliance to the target system is verifiable. Thus, in this experiment, we have selected one optimum solution from the extracted Pareto front of the learning phase and verified its generated rules. Due to the uniform partitioning of four input variables to five fuzzy sets, our ENF-S generates 625 rules. Therefore, we have shown the most frequently fired fuzzy rules during the scheduling process of different application graphs in Fig. 6. In this figure, the first four columns are the input variables as the premise parts of the fuzzy rules (four objectives of our problem) and the last one is the output of FNN as fuzzy rules’ consequent part (degree of each core in task assignment process). In this figure, the color spectrum reveals the linguistic values from “very low” (blue) to “very high” (red) for various input variables and output. Linguistic values of different colors are shown in Fig. 3.

Results and Discussion. As this figure shows, the existing trade-offs among the main objectives of our target scheduling problem are satisfied by generated rules. Distribution in colors of input and output columns shows that our learning process extracts the rules that explore the design space appropriately. Furthermore, the extreme values that contradict the mentioned trade-off are not fired frequently hence they are not included in this figure. The effect of values of all input variables is reflected in the core degree column. Indeed, when the temperature, power consumption, and execution time are at high levels, this processing core is overloaded and is not a proper choice for task assignment, therefore its degree should be medium. Rule index 20 of Fig. 6 indicates this case, where the core’s criticality degree is shown in yellow (interpreted as "medium value") due to the high values of power consumption, execution time, and medium value of temperature (shown as red and yellow). However, the first rule of Fig. 6 (indexed as zero) indicates a proper case where only power is in the "medium" level and the other parameters are low and very low. In this case, the criticality degree of the core is shown in blue (interpreted as "low value") and could be a proper choice for the task assignment.

3) Comparison With Previous Methods: Description and Aims. In this section, the efficiency of our proposed method (ENF-S) is compared to several related scheduling approaches. These selected methods are considered from various types
Fig. 7. Comparison of ENF-S with related scheduling approaches in terms of: (a) Execution time (b) Temperature (c) Failure rate (d) Power consumption.

including ERPOT [5], PowerPerf-PET (PPPET) [20], FIS-based [24], GSA [23], EE-RTS, MA-RTS, LA-RTS [19] and NSGA-II-based [26]. These methods consider one or multiple objectives of our defined scheduling problem. They are categorized in heuristic, fuzzy-based and meta-heuristic approaches which their details are previously discussed (see Section II). Fig. 7 compares the performance of ENF-S to the selected methods in terms of execution time, temperature, failure rate, and power consumption. Since our learning process is unsupervised considering a sequential procedure (the task scheduling), this figure could also demonstrate the Accuracy of our proposed multi-objective scheduling approach over a non-real ideal case where non of the objectives changes during the execution. These comparisons are performed on test dataset graphs (Table II) consisting of real-world and synthetic applications. It should be noted that for multi-objective methods the average values of all distinct solutions of the Pareto front in various aspects are reported in this figure.

Results and Discussion. As Fig. 7 shows, our proposed method (ENF-S) performs efficiently in all objectives and application graphs. ENF-S (shown as blue bars in Fig. 7) mostly outperforms the related approaches in terms of all reported critical parameters. This result is expected due to the thorough exploration of the design space and considering various objectives in the learning phase of ENF-S that leads to extracting appropriate fuzzy rules. Based on Fig. 7(a) ENF-S has appropriate execution time in many cases. Its execution time is very close to MA-RTS that is a single objective approach that only considers performance. However, based on Fig. 7(b) to (d), ENF-S significantly outperforms MA-RTS in other criteria. The explained case can be similarly deduced for EE-RTS and LA-RTS as other single objective approaches that consider energy and lifetime reliability, respectively. Averagely, ENF-S has 19.21%, 13.07%, 25.09%, and 13.16% improvement in execution time, temperature, failure rate, and power consumption, respectively over the considered related studies for all regarded test application graphs. Moreover, these improvements are also estimated for training dataset as: 22.54%, 17.74%, 31.38%, and 15.96% in execution time, temperature, failure rate, and power consumption respectively. These results show the importance of considering all critical parameters in scheduling simultaneously.

ENF-S also outperforms multi-objective-based schedulers in many cases. PowerPerf-PET and GSA are multi-objective scheduling methods that do not extract the Pareto front. In PowerPerf-PET, one solution is extracted by summing up the objectives with various weights and in GSA the objectives are considered hierarchically. Based on Fig. 7, GSA mainly has proper capability in execution time and temperature but loses in terms of power consumption and failure rate in all applications. Furthermore, NSGA-II is a Pareto front-based approach that performs properly in most cases, but it needs to apply a population-based exploration of design space for each application graph which is a very time-consuming process. Finally, the FIS-based approach utilizes a static fuzzy inference system to jointly optimize the critical parameters. In this method, the assignment rules are determined statically based on the expert’s knowledge. It maps tasks to processing cores based on some kinds of biased voting toward the lower values among the main optimization objectives. Based on Fig. 7, our proposed ENF-S outperforms this FIS-based scheduling in all critical parameters due to its considered learning approach in determining assignment rules that adapt them to the system’s conditions.
To better analyse the efficiency of ENF-S, we have performed a statistical study on all critical parameters to compare it to the static FIS approach [24] and a baseline real-time scheduler that only considers deadline inspired by EDF [1] in Fig. 8. Here, the test dataset application graphs are considered, and the distribution of normalized values for each objective is compared. Based on this figure, ENF-S has appropriate performance in all criteria. Its improvement over FIS-based method [24] is due to its learning ability in generating optimal rules that considers the joint optimization of the critical parameters rather than the static rule set. Moreover, compared to EDF, which is a basic greedy and single objective scheduling approach considering execution time, ENF-S improves the objectives efficiently by considering their trade-offs.

4) Overhead Analysis of ENF-S: Since our proposed ENF-S is an online scheduler, it could enforce runtime overhead to the system. In this section, we aim at calculating the ENF-S’s overhead. ENF-S consists of two phases: 1) learning FNN, and 2) using FNN as an online scheduler. The first phase is an offline process that is performed once at design time based on investigating different FNNs for scheduling various training applications. Since this phase is performed at design time, it has no runtime overhead. The learned FNN is employed as a scheduler during the system execution. Based on Algorithm 2 along with (12) to (15) this FNN consists of “for loops” and simple arithmetic operations. Thus, if the application consists of n tasks, the architecture has C cores, and the FNN comprises R fuzzy rules, the time complexity of the scheduling would be $O(RCn)$. Since the number of fuzzy rules (R) and the number of cores (C) are fixed, the time complexity of the proposed method is $O(n)$.

Moreover, in our simulations, we measured the time, temperature, and power overhead of applying ENF-S to the system. In this context, we have considered mid-scaled random application graphs (see Table II) and present these parameters averagely. Based on our simulations, ENF-S enforces about 3.2%, 0.9%, and 1.1% overheads in execution time, power consumption, and temperature, respectively.

VI. CONCLUSION

In this paper, an evolutionary-neuro-fuzzy task scheduling approach (ENF-S) is proposed. ENF-S is a rule-based online scheduler that receives the current state of each core and calculates its criticality degree for executing the current task. Next, the current ready task is assigned to the core with the lowest value of criticality degree. Besides, considering power management, a proper frequency level on the selected core is determined regarding DVFS property. At design time, ENF-S is learned using a multi-objective evolutionary algorithm (NSGA-II) to jointly optimize the critical parameters of heterogeneous multicore systems, including execution time, temperature, failure rate, and power consumption. Here, uniform partitioning of the input space is used to adjust premise parts’ parameters of fuzzy rules. Moreover, NSGA-II is used to learn consequent parts’ parameters of fuzzy rules based on optimizing the critical parameters of multi-core systems. Afterward, this trained fuzzy neural network is employed as a scheduler at runtime.

To demonstrate the efficiency of the proposed method, several experiments on synthetic and real-world application graphs are performed. These experiments investigate the capability of ENF-S in meeting the existing trade-offs among various critical parameters. Moreover, the interpretability of ENF-S is shown by studying its extracted fuzzy rules. Further, the efficiency of ENF-S is compared with some related methods. Last, the overhead of ENF-S is analyzed. These experiments, show the efficiency of ENF-S in its various aspects and over previous models in all criteria. As the future trend, expanding the proposed method using higher-order fuzzy logic including the type-2 to better consider the uncertainty is proposed.

REFERENCES

[1] P. Marwedel, Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things. Berlin, Germany: Springer, 2021.
[2] J. Henkel and N. Dutt, Dependable Embedded Systems. Berlin, Germany: Springer, 2021.
[3] A. B. Abdallah, Advanced Multicore Systems-On-Chip. Berlin, Germany: Springer, 2017.
[4] G. Xie, X. Xiao, H. Peng, R. Li, and K. Li, “A survey of low-energy parallel scheduling algorithms,” IEEE Trans. Sustain. Comput., vol. 7, no. 1, pp. 27–46, First Quarter 2022.
[5] A. Abdi, A. Girault, and H. R. Zarandi, “ERPO: A quad-criteria scheduling heuristic to optimize execution time, reliability, power consumption and temperature in multicorecs,” IEEE Trans. Parallel Distrib. Syst., vol. 30, no. 10, pp. 2193–2210, Oct. 2019.
[6] H. Lv, Y. Guo, and L. Yang, “Study of energy-efficient scheduling in multicore systems in dynamic voltage and frequency adjustment,” J. Phys.: Conf. Ser., vol. 2033, IOP Publishing, 2021, Art. no. 012143.
[7] H. Yang, I. Bacivarov, D. Rai, J. J. Chen, and L. Thiele, “Real-time worst-case temperature analysis with temperature-dependent parameters,” Real-Time Syst., vol. 49, no. 6, pp. 730–762, 2013.
[8] Y. Ma, T. Chantem, R. P. Dick, and X. S. Hu, “Improving system-level lifetime reliability of multicore soft real-time systems,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 6, pp. 1895–1905, Jun. 2017.
[9] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, “The case for lifetime reliability-aware microprocessors,” ACM SIGARCH Comput. Architecture News, vol. 32, no. 2, 2004, Art. no. 276.
[10] Y. Ma, J. Zhou, T. Chantem, R. P. Dick, and X. S. Hu, “Resource management for improving overall reliability of multi-processor systems-on-chip,” Dependable Embedded Syst., Berlin, Germany: Springer, 2021, Art. no. 233.
[11] M. Karami, A. Abdi, and H. R. Zarandi, “A cross-layer aging-aware task scheduling approach for multiprocessor embedded systems,” Microelectronics Rel., vol. 85, pp. 190–197, 2018.
[12] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li, “Energy-efficient scheduling for real-time systems based on deep Q-learning model,” IEEE Trans. Sustain. Comput., vol. 4, no. 1, pp. 132–141, First Quarter 2019.
[13] Y. Ma, J. Zhou, T. Chantem, R. P. Dick, S. Wang, and X. S. Hu, “Online resource management for improving reliability of real-time systems on “big–little” type MPSoCs,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 39, no. 1, pp. 88–100, Jan. 2020.

[14] A. Abdi and H. R. Zarandi, “HYSTERY: A hybrid scheduling and mapping approach to optimize temperature, energy consumption and lifetime reliability of heterogeneous multiprocessor systems,” J. Supercomput., vol. 74, pp. 2213–2238, Jan. 2018.

[15] M. Ansari et al., “Thermal-aware standby-sparing technique on heterogeneous real-time embedded systems,” IEEE Trans. Emerg. Topics Comput., vol. 10, no. 4, pp. 181–189, Fourth Quarter 2017.

[16] P. V. de Campos Souza, “Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature,” Appl. Soft Comput., vol. 92, 2020, Art. no. 106275.

[17] M. M. Ebadzadeh and A. Salimi-Badr, “IC-FNN: A novel fuzzy neural network with interpretable, intuitive, and correlated-contours fuzzy rules for function approximation,” IEEE Trans. Fuzzy Syst., vol. 26, no. 3, pp. 1288–1302, Jun. 2018.

[18] H. Topcuoglu, S. Hariri, and M.-Y. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3, pp. 260–274, Mar. 2002.

[19] J. Zhou, K. Cao, J. Sun, Y. Zhang, and T. Wei, “A framework to solve the energy, makespan and lifetime problems in reliability-driven task scheduling,” in Proc. Int. Conf. Internet Things IEEE Green Comput. Commun. IEEE Cyber Phys. Social Comput. IEEE Smart Data, 2019, pp. 608–614.

[20] H. F. Sheikh and I. Ahmad, “Sixteen heuristics for joint optimization of performance, energy, and temperature in allocating tasks to multi-cores,” ACM Trans. Parallel Comput., vol. 3, no. 2, pp. 1–29, 2016.

[21] B. Salami, H. Noori, and M. Naghibzadeh, “Fairness-aware energy efficient scheduling on heterogeneous multi-core processors,” IEEE Trans. Comput., vol. 70, no. 1, pp. 72–82, Jan. 2021.

[22] J. C. Da Silva, L. Leão, V. Petrucci, A. Gamatié, and F. M. Pereira, “Mapping computations in heterogeneous multicore systems with statistical regression on inputs,” in Proc. IEEE IX Braz. Symp. Comput. Syst. Eng., 2020, pp. 1–8.

[23] G. Bhat, G. Singla, A. K. Unver, and U. Y. Ogras, “Algorithmic optimization of thermal and power management for heterogeneous mobile platforms,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 3, pp. 544–557, Mar. 2018.

[24] Z. Ekhityari, V. Moghaddas, and H. Beitolhali, “A temperature-aware and energy-efficient fuzzy technique to schedule tasks in heterogeneous MPSoC systems,” J. Supercomput., vol. 75, pp. 5398–5419, Mar. 2019.

[25] M. Akbari, H. Rashidi, and S. H. Alizadeh, “An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems,” Eng. Appl. Artif. Intell., vol. 61, pp. 35–46, 2017.

[26] A. Abdi and H. R. Zarandi, “A meta heuristic-based task scheduling and mapping method to optimize main design challenges of heterogeneous multiprocessor embedded systems,” Microelectron. J., vol. 87, pp. 1–11, 2019.

[27] S. K. Roy, R. Devaraj, A. Sarkar, K. Maji, and S. Sinha, “Contention-aware optimal scheduling of real-time precedence-constrained task graphs on heterogeneous distributed systems,” J. Syst. Archit., vol. 105, 2020, Art. no. 101706.

[28] S. K. Roy, R. Devaraj, A. Sarkar, and D. Senapatii, “SLAQA: Quality-level aware scheduling of task graphs on heterogeneous distributed systems,” ACM Trans. Comput. Syst., vol. 20, no. 5, pp. 1–31, 2021.

[29] S. B. Maurer, “Directed acyclic graphs,” Handbook of Graph Theory. Boca Raton, FL, USA: CRC Press, 2013, pp. 180.

[30] R. Dick, “Embedded system synthesis benchmarks suite (E3S).” Accessed: Feb. 15, 2023. [Online]. Available: https://erdikr4.eecs.umich.edu/~rickp/e3s/

[31] M. Walker, S. Bischoff, S. Diestelhorst, G. Merrett, and B. Al-Hashimi, “Hardware-validated CPU performance and energy modelling,” in Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw., 2018, pp. 44–53.

[32] S. H. Voldman, M. Walker, S. Bischoff, S. Diestelhorst, G. Merrett, and B. Al-Hashimi, “MiBench: A free, commercially representative embedded benchmark suite,” in Proc. IEEE 4th Annu. Int. Workshop Workload Characterization, 2001, pp. 3–14.

[33] C. Biema, S. Kumar, J. P. Singh, and K. Li, “The parsec benchmark suite: Characterization and architectural implications,” in Proc. Int. Conf. Parallel Architectures Compilation Techn., 2008, pp. 72–81.

[34] S. Yari-Karim, R. Siyadzadeh, M. Ansari, and A. Ejiali, “Passive primary/backup-based scheduling for simultaneous power and reliability management on heterogeneous embedded systems,” IEEE Trans. Sustain. Comput., early access, 2022, doi: 10.1109/TSCUS.2022.3186656.

[35] “Task graphs for free (TGFF).” Accessed: Feb. 15, 2023, [Online]. Available: https://robertdick.org/projects/tgff/

Athena Abdi received the BSc, MSc, and PhD degrees in computer engineering all from the Amirkabir University of Technology, Tehran, Iran. She spent a year as a visiting researcher with the SPADES Team, INRIA Rhone-Alpes, France. Currently, she is an assistant professor with the Faculty of Computer Engineering, K. N. Toosi University of Technology. Her research interests include embedded systems, scheduling, and fault-tolerant design in computer architecture.

Armin Salimi-Badr (Senior Member, IEEE) received the BSc, MSc, and PhD degrees in computer engineering all from the Amirkabir University of Technology, Tehran, Iran, and the PhD degree in neuroscience from the University of Burgundy, Dijon, France. Currently, he is an assistant professor with the Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran. His research interests include computational intelligence and robotics.