An Energy-Aware Approach to Design Self-Adaptive AI-based Applications on the Edge

Alessandro Tundo*, Marco Mobilio*, Shashikant Ilager†
Ivona Brandić†, Ezio Bartocci†, Leonardo Mariani*

*University of Milano-Bicocca
Milan, Italy
{name.surname}@unimib.it

†Vienna University of Technology
Vienna, Austria
{name.surname}@tuwien.ac.at

***Abstract*—The advent of edge devices dedicated to machine learning tasks enabled the execution of AI-based applications that efficiently process and classify the data acquired by the resource-constrained devices populating the Internet of Things. The proliferation of such applications (e.g., critical monitoring in smart cities) demands new strategies to make these systems also sustainable from an energetic point of view.**

In this paper, we present an energy-aware approach for the design and deployment of self-adaptive AI-based applications that can balance application objectives (e.g., accuracy in object detection and frames processing rate) with energy consumption. We address the problem of determining the set of configurations that can be used to self-adapt the system with a meta-heuristic search procedure that only needs a small number of empirical samples. The final set of configurations are selected using weighted gray relational analysis, and mapped to the operation modes of the self-adaptive application.

We validate our approach on an AI-based application for pedestrian detection. Results show that our self-adaptive application can outperform non-adaptive baseline configurations by saving up to 81% of energy while loosing only between 2% and 6% in accuracy.

***Index Terms*—self-adaptive, energy-aware, AI-based, multi-objective, edge computing, internet-of-things**

I. Introduction

Both academia and industry raised the issue of the massive amount of energy consumed by ICT services and the rising energy costs [1]–[4]. Reducing energy consumption is a high priority objective, to wisely use the available resources. Indeed, building *sustainable AI-based applications* is a key technical challenge that engineers are facing nowadays [5].

AI-based applications are increasingly deployed on the *edge*, within resource-constrained environments that cannot indefinitely supply a constant amount of power, such as, battery-powered devices and computing nodes powered by renewable energy sources (e.g., photovoltaic panels or wind turbines) [6]–[8]. Such applications are particularly resource-intensive, thus, carefully using energy is a key requirement to feasibly run AI services within these environments. For example, critical monitoring services for smart cities (e.g., pedestrian detection and traffic analysis [9]–[11]), environmental monitoring applications (e.g., wildfire detection [12], [13], and wildlife monitoring [14], [15]), all require fast data processing and high accuracy, with cost-effective energy consumption.

These scenarios require consuming a large volume of data generated from Internet-of-Things (IoT) sensors in various forms (e.g., time series values, video streams, images) with resource-greedy machine learning models (e.g., exploiting TPUs or GPUs) [5], [16], [17]. In contrast, the feasibility of scenarios that involve battery-powered devices [18], [19] depends on the capability of reducing energy consumption to extend the battery life.

In response to this urge, researchers have investigated several approaches to design systems with *a controllable and programmable trade-off among quality, efficiency, and energy consumption*.

Energy-awareness and efficiency research mainly targets low-level tasks such as scheduling and provisioning [20]–[24], routing [25], data storage and processing [26], and machine learning models optimization [27]. Although valuable, only optimizing the low-level tasks may result in hardly-predictable performance of the applications. Thus, it becomes challenging or even impossible to balance competing application-level objectives (e.g., accuracy, energy consumption, and efficiency) working only on low-level features.

Other approaches targeted code optimizations [28], analysis of software energy consumption [29], [30], and architectural tactics to contain energy utilization [31] and costs [32]. Analyzing energy consumption retrospectively to take corrective actions (e.g., code or architectural refactoring) can be expensive and difficult to control in the long term.

In contrast with previous work, in this paper we investigate the challenge of configuring (e.g., determining the frame rate, the image resolution, and the kind of machine-learning model that an object detection system must use) AI-based edge applications, to *balance* energy consumption and application objectives. Despite this being a common challenge to any edge application, we target AI-based applications since they are frequently used in the edge, despite being resource-demanding.

Naively, we could hypothesize to simply systematically and exhaustively explore the configuration space of an application, and then determine the best configuration to use. In practice, there are two main obstacles: the *huge cost of the exploration*

of the configuration space and the *lack of a configuration that globally optimizes every objective*.

Exploring the configuration space of AI-based edge applications is extremely expensive due to the *size* of the space, determined by the high number of configuration parameters and parameter values, and the *cost of sampling*, which requires running multiple experiments, to determine how much a configuration fulfills the energy and application objectives [33]. This cost is even higher in large distributed and heterogeneous environments, where different nodes or groups of nodes may require to be optimized individually.

Furthermore, *different run-time scenarios usually require different configurations* to be addressed properly. For instance, detecting objects in situations where the objects to be detected occur rarely (e.g., detecting pedestrians at night in a peripheral city area) is completely different from detecting the same objects in situations where the objects occur densely and repeatedly (e.g., detecting pedestrians in an area near a stadium after a concert). Thus, no single configuration can optimize both accuracy and energy consumption in all circumstances, but applications need to adapt to changing conditions to behave optimally.

We address these challenges by proposing an *energy-aware* approach that can guide developers to implement an *AI-based self-adaptive application* able of switching its operation modes in response to changes in the environment, finally balancing energy consumption with the application-level objectives.

In a nutshell, this work provides the following contributions.

An energy-aware approach for the design of AI-based self-adaptive applications. We present an approach to design and implement an AI-based self-adaptive application that can dynamically balance application requirements and energy consumption, according to a behavioral model derived empirically.

A meta-heuristic search procedure combined with a weighted configuration extraction process. We define a meta-heuristic search procedure that allows to empirically sample a tiny portion of the configuration search space, to finally extract, using *weighted gray relational analysis*, a set of configurations that correspond to the operation modes employed by the self-adaptive system.

A smart city scenario prototype implementation. We showcase the applicability of the proposed approach by implementing the prototype of an AI-based self-adaptive application for a pedestrian detection scenario involving a single-board computer equipped with a camera and a hardware accelerator (i.e., an Edge TPU).

Empirical evidence of the effectiveness of the approach. We answer two research questions by performing in-lab experiments and evaluating pedestrian detection scenarios following real-word pedestrian traffic shapes. Results show that configurations obtained through the meta-heuristic search procedure perform comparably well with respect to the ones obtained by a near-exhaustive search of the space. The comparison to four non-adaptive baseline applications shows that the self-adaptive system is able to self-adapt its operation mode to the pedestrian traffic shapes saving up to 81% of energy consumption. At the same time, it guarantees a similar accuracy when compared to the most accurate configurations, losing between 2% and 6% only, but outperforming 3 out of 4 non-adaptive applications on the processing speed gaining between 77% and 233%.

The paper is organized as follows. Section II presents a Smart Traffic Monitoring (STM) motivational scenario. Section III describes our approach, with specific reference to the motivational scenario. Section IV presents the empirical results. Section V discusses related work. Finally, Section VI presents concluding remarks and future work.

II. Motivational Scenario

According to the latest report released by Governors Highway Safety Association (GHSA), nearly 3.500 pedestrians died in the United States in the first six months of 2022 (+5% from the same period in 2021) [34]. In three years, pedestrian deaths raised about 18%, that is, nine times faster than U.S. population growth [35]. Similarly, the European Transport Safety Council (ETSC) reported 20.600 road deaths in the EU last year, with vulnerable road users (pedestrians, cyclists, and users of powered two-wheelers) representing just under 70% of total fatalities within urban areas [36], [37]. Addressing this critical issue of preventing accidents not only depends on social education [38] but also requires developing Smart Traffic Monitoring (STM) systems that enable digital monitoring of urban traffic [39]–[41], real-time analytics [42], [43], and intelligent driver assistants [9], [44], [45].

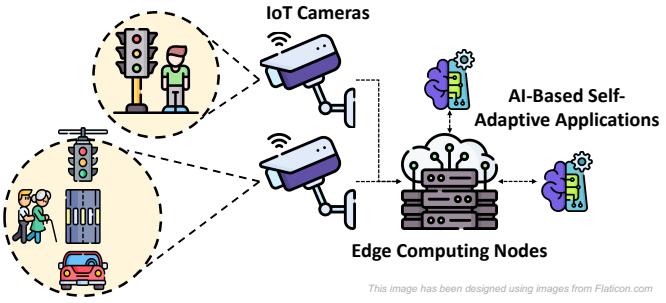


Fig. 1: A pedestrian detection scenario.

An STM system requires continuous monitoring of the traffic scenarios to identify potential incidents (e.g., the presence of pedestrians in blind spots) through video streams and processing frames, and alerting the nearby vehicles through the use of 5G-enabled edge nodes [9]. Such an STM system can host hundreds of cameras and sensors deployed to roads in cities and countryside areas [46].

The edge devices processing video streams are in always-on mode and potentially powered by batteries or renewable energy sources at the edge, which is the basis for limited and unreliable power supply. Hence, reducing energy consumption and executing critical emergency applications become extremely important. On the other hand, such critical applications expect a minimum QoS for safety and reliability (e.g., inference time and ML model accuracy). Therefore, they require continuous monitoring of resources (e.g., energy budget) and workload

(e.g., number of detected pedestrians in time intervals), and when needed, employing self-adaptive applications and adapting hardware and software configurations (e.g., camera resolution, ML model, and hardware acceleration).

Figure 1 depicts a pedestrian detection scenario where an application can employ different operation modes according to pedestrian traffic volumes. For instance, this scenario could be addressed with four operation modes as defined in Table I. A self-adaptive application for this scenario can autonomously balance resource (e.g., energy consumption) and application requirements (e.g., frame processing speed and accuracy) by switching among the different operation modes.

On the contrary, using a single operation mode for a whole day cannot adapt to a changing environment. Considering a smart-city scenario with hundreds of IoT cameras and dozens of application instances deployed across several edge nodes, the benefits of such an approach are exponential.

III. An Approach to Design Energy-Aware Self-Adaptive Applications

A *self-adaptive application (SAA)* is an application capable of modifying itself or other connected resources in response to a continuously changing operational environment [47], [48].

A SAA consists of a pair (AL, MR), where AL is the *adaptation logic*, and MR represents the *managed resources* [49], which are a group of resources, such as robotics, vehicles, and generic hardware with software, that the SAA can control [49]. The adaptation logic is composed by all those items responsible for monitoring the environment (M), analyzing the data (A), planning (P), and executing the adaptation (E). This basic feedback framework proposed by Kephart and Chess [50] is named MAPE loop, and it is often extended by a knowledge component (K) responsible for managing content (e.g., monitoring values and adaptation policies).

SAAs are particularly effective in resource-constrained environments. We consider here the case of an AI-based application that implements the pedestrian detection use-case described in Section II and that is hosted on an embedded device (e.g., a Raspberry Pi) equipped with a video camera and a hardware accelerator (e.g., a TPU). The device executes an application capturing frames from the camera and processing them with an object detection model to detect pedestrians.

The hardware accelerator boosts the processing speed by lowering the ML model inference time. In this context, we must consider three main objectives: achieving high detection accuracy, processing frames at a high rate, and reducing energy consumption.

Optimizing these objectives at the same time for every possible operational condition is generally infeasible. Interestingly, a SAA can dynamically balance the degree of satisfaction of these objectives depending on the run-time context. However, engineers designing SAAs need to identify *suitable configurations* for the run-time to balance the chosen objectives. Further, SAAs have to implement the logic to automatically switch between configurations (e.g., the four operation modes reported in Table I), to adapt to changes in the operational environment (e.g., the pedestrian traffic volumes).

Identifying the configurations that implement the intended operation modes is also challenging, especially for AI-based applications running on heterogeneous and resource-constrained nodes. Indeed, simply using a simulator may lead to results largely diverging from the real behavior of these applications. On the other hand, taking empirical measures by running the real devices and applications can be extremely expensive, especially when large configuration spaces must be explored [33]. We propose here an approach that combines the benefits of the *empirical identification* of the configurations and those of an intelligent *exploration of the configuration space* to yield suitable solutions to design an effective and energy-aware SAA.

Figure 2 describes our approach with a workflow diagram. An engineer provides the adaptation logic (A) as a finite-state machine (FSM) whose states represent the SAA operation modes and whose transitions encode the switching conditions between them. In parallel, the engineer identifies the configuration space to explore, and defines a Multi-Objectives Optimization Problem (MOOP) that can be solved automatically (B) using a meta-heuristic search procedure. Furthermore, the engineer specifies weights and thresholds for the objectives to guide the (C) extraction of the configurations to set in each operation mode. The workflow terminates (D) with the implementation of the final FSM.

In the next subsections, we describe each step of the workflow in detail and exemplify the approach with the pedestrian detection scenario described in Section II.

A. Defining the State-Based Adaptation Logic

The first step of our approach requires an engineer, supported by domain experts, to define, in a rigorous way, the *behavioral model* of the self-adaptive application [51].

As specification we use a *Finite-State Machine (FSM)*, since it allows to explicitly represent the adaptation logic of an SAA [52]–[54]: the states represent the operational modes of the SAA, and the transitions represent the conditions triggering a change in the operation mode of the application.

Formally, an FSM M is defined by a tuple (S, Σ, δ, s_0), where S is the set of states, Σ is the set of the input symbols, that is, the set of events that may trigger state transitions, δ is the set of all the possible transitions from a state $s_1 \in S$ to a state $s_2 \in S$ caused by an event $\sigma \in \Sigma$, s_0 is the initial state.

Let us consider the pedestrian detection scenario again. Here an engineer may want to define a SAA that can self-adapt across four operation modes (see Table I) to address the four possible run-time contexts in the area where the camera shall be deployed, defined for instance according to the available studies [46], [55], [56]. Each operation mode, for example *low-energy*, represents the working condition of the software that is best suited for the corresponding run-time context, for example *few pedestrians detected*. Each operation mode must satisfy certain characteristics in terms of energy consumption,

TABLE I: A set of four operation modes used in our motivational pedestrian detection scenario.

Operation Mode	Runtime Context	Desirable Characteristics		
		Energy Consumption	***Detection Accuracy***	***Frames Processing Rate***
power-saving	no pedestrians detected	very low	low	moderate
low-energy	few pedestrians detected	low	moderate	moderate
high-accuracy	small group of pedestrians detected	moderate	high	high
high-rate	crowd detected	high	moderate	very high

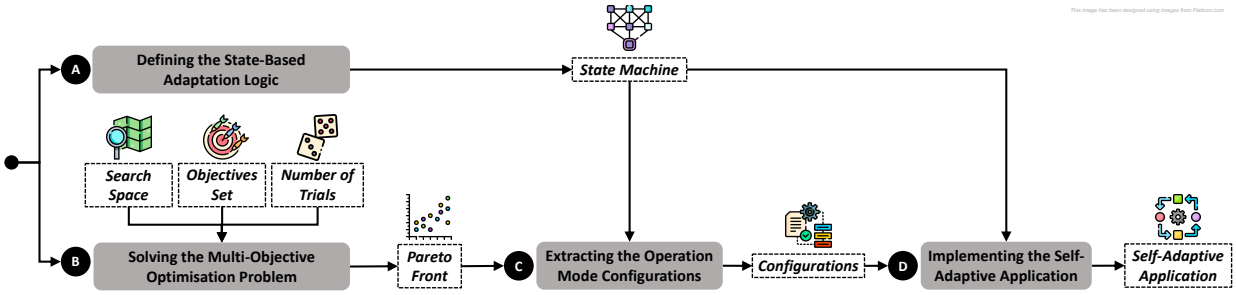


Fig. 2: The steps of the proposed approach represented as a workflow diagram.

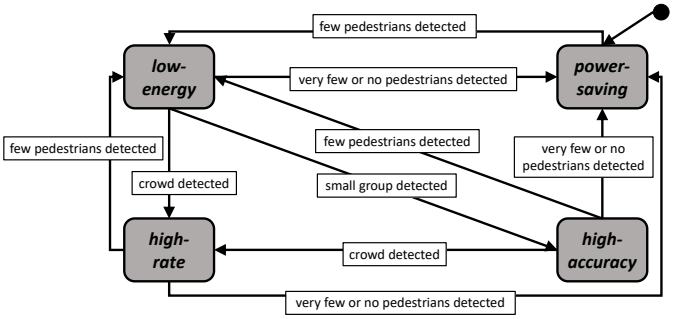


Fig. 3: An abstract state machine modeling the states and the transitions of a self-adaptive application for our scenario.

detection accuracy and frames processing rate. These characteristics are used to identify the exact software configurations at step (C) Extracting the Operation Mode Configurations by providing the corresponding sets of objective weights and thresholds. Figure 3 shows an abstract FSM, with the four identified abstract states and 9 transitions that capture when the software must self-adapt. Please note that the domain-knowledge is exploited here to determine the transitions that must be encoded in the FSM, among the full set of the possible state transitions.

B. Solving the Multi-Objective Optimization Problem

Finding high-quality software configurations that correspond to the operation modes identified by the engineer (e.g., the four states shows in Figure 3) is a hard problem. AI-based applications can be configured according to several parameters (see for instance the list of parameters that may influence pedestrian detection listed in Table II), generating a huge exploration space that cannot be exhaustively explored. Computer-simulated experiments can reduce the time and effort, but they are usually inaccurate, especially in Cyber Physical Systems and other domains that include real-world metrics [57]. To address this challenge, we defined a *Multi-Objective Optimization Problem (MOOP)* that is able to discover the configurations that deliver the best results for the considered set of objectives, and that can be exploited to find the actual configurations that effectively implement the operation modes represented as states of the FSM.

An optimization process aims to find a set of input values for a problem to obtain the "*optimal*" output values. The definition of optimality is problem-specific, and formally, it refers to minimizing or maximizing one or more objective functions by varying the input values. Hence, a MOOP requires the satisfaction of a number of different and often conflicting objectives at the same time [58], [59]. Intuitively, there is no single best solution for all the objectives, but rather there exist several optimal solutions representing the best trade-offs among all the objectives [58]. The set of all possible solutions constitutes the *search space*, which then also contains the set of input values revealing optimal outputs.

We define the search space X as a set *configurations*. A configuration *conf* is n-tuple $(c_1, \ldots, c_n)$, where c_k is the value of the k-th configurable parameter $p_k \in P$ assuming values in its domain D_{p_k}. The size of X is $|X| = \prod_{k=1}^{n} |D_{p_k}|$.

The set of solutions X^* is called the Pareto front, which contains all the solutions where no improvement is possible in any objective function without sacrificing at least one of the other objective functions [59]. This is also referred to as the non-dominated solutions set.

In the pedestrian detection scenario we have three objectives: (i) maximize the pedestrians detection accuracy (*acc*), (ii) minimize the energy consumption (*eng*), and (iii) maximize the number of processed frames in a time window (*rate*). Hence, we define a MOOP with these three objectives (depending on the specific case, we might have a different number

of objectives):

$$\begin{aligned} \min \quad & -acc(conf) \wedge eng(conf) \wedge -rate(conf) \\ \text{s.t.} \quad & conf \in X \end{aligned} \tag{1}$$

The search space X is defined as a set of configuration quintuples with five configuration parameters for our application, that is, the camera resolution (R), the camera frame rate (FPS), the object detection model (M), the detection threshold (T), and whether to use the external hardware accelerator (TPU). Each parameter domain has a different cardinality (see details in Table II). Accordingly, $|X| = |R| \times |FPS| \times |M| \times |T| \times |TPU| = 3402$ configuration quintuples.

Solving the Eq. 1 results in a Pareto front with non-dominated solutions, that is, configurations that *fulfill the three objectives by a different, but relevant, degree*. We use a strategy derived from NSGA-II to compute the Pareto front.

NSGA-II is a solid, fast, and widely used optimization algorithm in real-world applications [60]. We use the approach defined by Deb et al. [61] for the exploration of the search space: it is explored by searching for dominant solutions (i.e., the fitness of a solution is defined by computing its non-domination level) in less populated areas of the space (i.e., determined by computing the crowding distance) guaranteeing the diversity of the identified solutions; mutations randomly change parameter values with a probability that is computed according to the number of parameters in the configuration, and uniform crossover recombines configurations with a probability of 0.9.

During the search space exploration, our procedure records all the evaluated objective values, and at the end it extracts the Pareto front from the whole results set. In the empirical evaluation, we show how this strategy can be used to explore only 10% of the search space to select nearly optimal configurations. Note this is particularly relevant, since assessing how a single configuration fulfills the three objectives requires collecting empirical measures by repeating a same experiment multiple times.

C. Extracting the Operation Mode Configurations

The Pareto front obtained by solving the MOOP usually contains a large number of non-dominated solutions, compared to the operation modes needed by the self-adaptive application. The decision-making process to identify the actual solutions from the Pareto front involves comparing multiple criteria, trading-off certain objectives for others [62], [63]. To address this problem, we use the *weighted gray relational analysis (WGRA)* [62] method, a weighted version of the GRA introduced by Ju-Long [64] and employed in multiple application domains [65]. This is a very robust method [66], preferable to other multi-criteria decision making (MCDM) methods as it inherently incorporates uncertainty in data, and it is simple to calculate [66], [67] and to integrate into existing software.

GRA combines into a single value all the objectives. This simplifies the original MCDM problem into a single-criterion decision-making problem [62], making Pareto front solutions easily comparable. To let engineers extract states that fulfill the objectives by different degrees, we employ the weighted version of the algorithm that uses a set of weights W to give more importance to certain objectives [65].

The WGRA algorithm consists of three main steps: (i) data normalization, (ii) reference network computation, and (iii) gray relational grade (GRG) computation [63].

The *data normalization* step consists of the normalization of the objective values in the Pareto front according to two cases: larger-the-better for maximization, and smaller-the-better for minimization. The normalized value F_{ij} is calculated by Eq. 2 and 3 for maximization and minimization cases, respectively:

$$F_{ij} = \frac{f_{ij} - \min_{i \in n} f_{ij}}{\max_{i \in n} f_{ij} - \min_{i \in n} f_{ij}} \tag{2}$$

$$F_{ij} = \frac{\max_{i \in n} f_{ij} - f_{ij}}{\max_{i \in n} f_{ij} - \min_{i \in n} f_{ij}} \tag{3}$$

with f_{ij} as the i-th value of the j-th objective in the matrix O, a matrix $n \times m$ composed of n Pareto front solutions and m objectives. F_{ij} is the value of f_{ij} after normalization.

The *reference network computation* step consists in forming the reference network F_j^+, that is, an ideal network obtained by choosing the best value of each of the objectives as follows:

$$F_j^+ = \max_{i \in n} F_{ij} \tag{4}$$

Finally, the *gray relational grade (GRG) computation* step consists in calculating the similarity between each candidate network (i.e., the objective values of each optimal solution in the Pareto front) and the reference network F_j^+. The GRG for each i-th value in the Pareto Front is computed as follows:

$$GRG_i = \frac{1}{n} \sum_{j=1}^{m} w_j \frac{\Delta\min - \Delta\max}{\Delta_{ij} + \Delta\max} \tag{5}$$

where w_j is the weight of the j-th objective value (with $\sum_{j=1}^{m} w_j = 1$); $\Delta ij = |F_j^+ - F_{ij}|$ is the absolute value of the difference of between the j-th objective value in the reference network and the one in the candidate network; $\Delta\max = \max_{i \in n, j \in m}(\Delta ij)$ and $\Delta\min = \min_{i \in n, j \in m}(\Delta ij)$ are the maximum and minimum deltas, respectively.

The $conf \in X$ with the largest GRG_i is the recommended optimal solution outputed by the WGRA process. Depending on the set of weights used to extract the configuration from the Pareto front, the configuration shall map to a different state of the FSM, that is, it implements a different operation mode of the AI-based edge service.

To illustrate further, let us focus on two operation modes in our example, namely, *power-saving* and *high-rate*. The engineer, jointly with domain experts [68], may provide the following sets of weights for the two operation modes, respectively: $W_{\textit{power-saving}} = \{0.05, 0.9, 0.05\}$ and $W_{\textit{high-rate}} = \{0.6, 0, 0.4\}$. The specific weights could be derived from a Service Level Agreement (SLA) defining the QoS, and the costs the application service provider to sustain and deliver the application.

Engineers could also define a set of objective thresholds t_j for each objectives O_j to reduce the size of the Pareto front

TABLE II: The domain of the parameters used to define the search space of the multi-objective optimization problem.

Parameter	Parameter Type	Domain
Camera Resolution (R)	Categorical	{1920x1080, 1280x720, 640x480}
Camera Frame Rate (FPS)	Categorical	{1, 5, 10, 15, 20, 25, 30}
Object Detection Model (M)	Categorical	{SSD MobileNet V1, SSD/FPN MobileNet V1 TF2, SSD MobileNet V2, SSD MobileNet V2 TF2, SSDLite MobileDet, EfficientDet-Lite0, EfficientDet-Lite1, EfficientDet-Lite2, EfficientDet-Lite3}
Detection Threshold (T)	Numerical (low: 0.1, high: 0.9, step: 0.1)	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}
Use HW Accelerator (TPU)	Categorical	{true, false}

given in input to the WGRA algorithm, filtering out solutions that might be unreasonable for a given operation mode *op*. In particular, a solution is filtered from the Pareto front if the value it achieved on objective O_j is above the threshold t_j.

For example, let us consider the *power-saving* and the *high-rate* operation modes again. The weights assigned to the $W_{power\text{-}saving}$ set must give a large importance to the energy consumption objective in order to extract an energy-efficient configuration. However, this may lead to the identification of a very poor but still non-dominated solution for the other two objectives. To prevent this risk, the engineer can filter all the solutions that do not provide a minimum detection accuracy level and/or number of processed frames. For instance, they can define a set of thresholds $T_{power\text{-}saving} = \{t_{acc}, t_{eng}, t_{rate}\} = \{0.2, 0, 60\}$ to exclude solutions with a detection accuracy lower than 0.2, and a number of processed frames lower than 60. A completely different set of thresholds could be defined for the *high-rate*, that is, $T_{high\text{-}rate} = \{0.3, 0, 0\}$. In this case, solutions with a detection accuracy lower than 0.3 are filtered out in order to provide a minimum detection accuracy level, when compared to the *power-saving* mode.

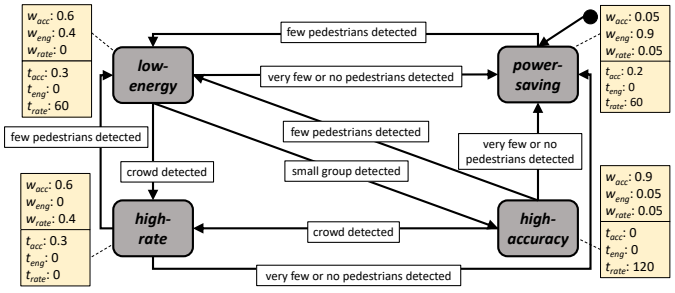


Fig. 4: A refined version of the abstract state machine shown in Figure 3 with the set of weights and thresholds for each of the operation modes.

Figure 4 shows the refined version of the abstract FSM previously shown in Figure 3 with the weights and thresholds for WGRA analysis defined by the engineers attached to states. The chosen weights and thresholds represent the actual specification of the *desirable characteristics* of the operation modes listed in Table I. The execution of the WGRA algorithm for each of the FSM state extracts a configuration $conf_{op}$ with the actual configuration parameter values that can be used by the SAA application to self-adapt the operation mode.

D. Implementing the Self-Adaptive Application

In the last step, the engineer is required to implement the self-adaptive application according to the output of the analysis. The abstract state machine is transformed into a concrete one in two steps: first, each of the transitions must be turned into an actual triggering condition; second, the operation mode configurations extracted in the previous step are mapped into a piece of logic able to set these configurations at runtime. Fig. 5 shows the final FSM for our pedestrian detection scenario, with actual conditions and operation modes.

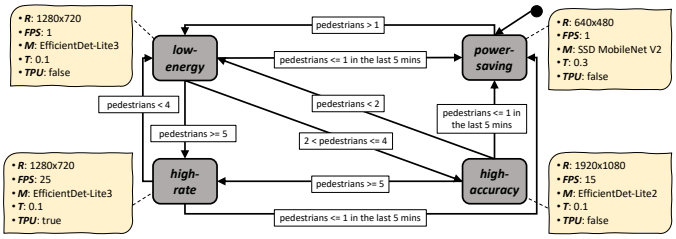


Fig. 5: The concrete finite state machine implementing a self-adaptive application for our scenario.

The FSM can be translated into working code using generators [69], [70] or when this is not possible or too difficult [71], the SAA can be obtained semi-automatically or manually [71]–[73]. Our approach outputs a concrete FSM encoding the SAA and does not bind the engineer to use any specific method to implement the SAA.

IV. Empirical Evaluation

To evaluate our approach, we investigate the following two research questions in the context of the pedestrian detection case study described in Section II. We select such a case study since it represents a real-world and challenging scenario that requires delivering effective and sustainable AI edge services. **RQ1 (Meta-Heuristic VS Near-Exhaustive Search) - Can our meta-heuristic search approach discover solutions whose quality is comparable to those obtained by a near-exhaustive search?** This research question investigates the effectiveness of our meta-heuristic strategy. In particular, it studies whether the heuristic exploration of a small portion of the search space can lead to results comparable to a near-exhaustive exploration.

RQ2 (Objectives Trade-Off) - Can a self-adaptive pedestrians detection application better balance energy consumption and application objectives compared to a non-adaptive application? This research question investigates whether the self-adaptive application resulting from our methodology can release a better trade-off among accuracy, energy, and processing speed compared to four baseline non-adaptive applications.

A. Experimental Setting

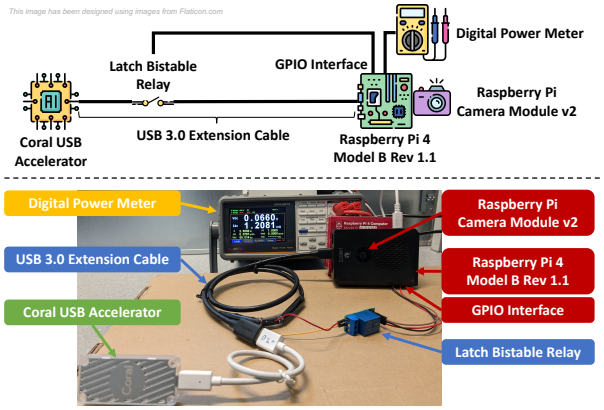


Fig. 6: The test-bed used to run the evaluation experiments.

Fig. 6 shows the test-bed we used to run our case study evaluation, first schematically (above), then its concrete in-lab implementation (below).

We employ a Raspberry Pi (RPi) 4 Model B Rev 1.1 (64-bit quad-core ARMv8, 4GB of RAM, RPi OS Lite 64-bit Debian GNU/Linux 11) equipped with the RPi Camera Module v2 and boxed in a LABISTS case[1] with a 5V fan connected to the RPi General Purpose Input/Output (GPIO) interface. The RPi is powered by a USB-C AC adapter connected through a GW Instek GPM-8213 digital power meter[2] that we use to collect instant power values.

To reduce the idle energy consumption of RPi, we disable the unnecessary components: all the LEDs (i.e., activity, power, and Ethernet port), the Wi-Fi antenna, the Bluetooth, and the HDMI port. Internet and private network connectivity is provided via network cable. A Coral USB Accelerator (Edge TPU)[3] is plugged-in for those experiments that require hardware accelerator. The accelerator is automatically powered-on when connected to the USB port.

Since there is no possibility to enable and disable a single USB port on-the-fly via software, a self-adaptive application running on such device would not be capable to completely power-off the accelerator when not in use, reducing the potential benefits of switching to an energy-efficient operation mode. To overcome this limitation, we realize a software-level power switch by employing a latch bi-stable relay (SONGLE SRD-05VC-SL-C) connected to the GPIO interface and a USB 3.0 extension cable. This enables us to turn it on and off by triggering the relay through software to close or open the circuit using a GPIO pin. For pedestrian detection, we employ state-of-art object detection models pre-trained on the COCO dataset [74]. The models are publicly available at the Coral.ai website[4], and they are already compiled for both CPU and Edge TPU execution. We reported detailed information to re-create our test-bed on our public repository https://gitlab.com/sustainable-continuum-monitoring/self-adaptive-moop/-/tree/ASE_2023?ref_type=tags.

B. RQ1 - Meta-Heuristic VS Near-Exhaustive Search

This research question aims to investigate whether exploring a small portion of the search space efficiently can lead to comparable results with a near-exhaustive exploration.

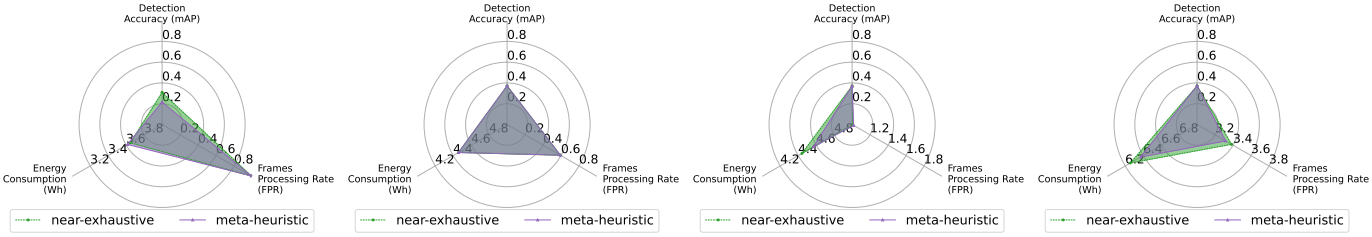
To answer RQ1, we first compute the Pareto front of the MOOP as defined in Eq. 1 with our meta-heuristic search procedure by only exploring 10% of the search space reported in Table II (i.e., 340 unique trials out of 3402 trials), and then we explore more than 80% of the same space (i.e., 2790 unique trials out of 3402 trials) with a random search procedure. Second, we extract the four operation modes needed to address the pedestrian detection scenario (according to the weights and thresholds reported in Fig. 4) from the two Pareto fronts: the one computed with the meta-heuristic search procedure and the one obtained with the near-exhaustive procedure. Finally, we compare the objective values achieved with the two SAAs that derive from the two sets of selected states. A good meta-heuristic procedure should be able to achieve results as good as the near-exhaustive exploration.

The whole optimization procedure is implemented with the Optuna framework [75], a state-of-art hyperparameter optimization framework with MOOP capabilities. Our meta-heuristic search procedure with memory capabilities is realized by using the NSGAIISampler[5] implementing the NSGA-II algorithm and the results database provided by Optuna. We use the default framework values to configure the sampler and we repeat the search 10 times with a different seed value recorded for reproducibility. The near-exhaustive search procedure, instead, employs the RandomSampler[6].

At each optimization round, when a sampler selects a point $conf$ from the search space, two experiments must be executed to determine the objective values for the selected $conf$.

The first experiment computes the *detection accuracy* by employing a pedestrian street scene belonging to the Multiple Object Tracking benchmark dataset [76] (i.e., the ADL-Rundle-6 video). Both the frame size and the ground truth have been properly adjusted to match the camera resolution (R) parameter values defined by the search space. We use the Mean Average Precision (mAP) as detection accuracy metric, a popular metric for object detection algorithms [77], and

[1]https://labists.com/products/raspberry-pi-4-case-kit
[2]https://www.gwinstek.com/en-GB/products/detail/GPM-8213
[3]https://coral.ai/products/accelerator
[4]https://coral.ai/models/object-detection/
[5]https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.NSGAIISampler.html
[6]https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.RandomSampler.html



(a) power-saving (b) low-energy (c) high-accuracy (d) high-rate

Fig. 7: Radar charts comparing the objective values of the four self-adaptive operation modes when employing a solution obtained with the meta-heuristic search procedure and one obtained with the near-exhaustive search procedure. The solutions are extracted with the WGRA method using the same set of weights and thresholds.

we evaluate the model predictions by using the open-source FiftyOne COCO-style evaluator[7].

The second experiment, instead, computes both the achieved *Frames Processing Rate* (FPR) and the *energy consumption*. We run the pedestrian detection application on the device (i.e., the Raspberry Pi described in Section IV-A for 120 seconds configured according to *conf*). We collect both the consumed energy in Watt-hours (Wh) and the FPR computed as the ratio between the number of processed frames and the experiment duration.

Results: The near-exhaustive search executed for about 18 days sampling 2790 unique trials and discovered a Pareto front with 131 solutions. The meta-heuristic search executed for about 54 hours sampling 340 unique trials (10% of the entire space) and discovered a Pareto front with 83 solutions on average. Note that the saving, when the sampling involves running experiments, is significant in both relative and absolute terms (more than 2 weeks of computing saved).

Since each run of the meta-heuristic search may return a slightly different configuration for a given state, we selected the configuration that occurred most frequently in the 10 repetitions to derive the corresponding SAA. When multiple solutions have the same highest frequency, we excluded the solution matching the one extracted from the near-exhaustive Pareto front to avoid any bias, and consider a worst case scenario.

Fig. 7 shows four radar charts - one per each operation mode in the SAA - comparing the three objective values of the solution extracted with the near-exhaustive search (green, dashed, dot mark), and the one extracted from meta-heuristic search (purple, solid line, triangle mark), respectively. Each of the axes has its own scale, but for all the objectives, the higher is the value the better it is.

The plots clearly indicate that the states identified by our meta-heuristic search procedure and the ones obtained with the near-exhaustive search result in highly similar performance. The *low-energy* operation mode (Fig. 7b) resulted in exactly the same solution returned by the two procedures. While the near-exhaustive search identified solutions performing comparably to the ones identified by the meta-heuristic search in the remaining three operation modes.

In the case of the *power-saving* operation mode (Fig. 7a), the two solutions perform with the same FPR and with negligible difference in energy consumption ($< 1\%$). The difference is slightly larger for the detection accuracy (0.307 mAP VS 0.215 mAP), whose relevance in the power-saving mode is however limited.

In the case of the *high-accuracy* operation mode (Fig. 7c), the two solutions perform with the same detection accuracy, and with negligible differences for FPR($< 1\%$) and energy consumption (4.442 Wh VS 4.570 Wh).

Finally, the two solutions obtained for *high-rate* (Fig. 7d) perform with the same detection accuracy, and with negligible differences for both FPR and the energy consumption ($< 2\%$).

We can conclude that our search procedure has been as effective as the near exhaustive procedure for the pedestrian detection scenario, despite an empirical exploration of only 10% of the search space.

C. RQ2 - Objectives Trade-Off

This research question aims to investigate whether a self-adaptive application changing its operation mode can better balance the fulfillment of multiple objectives compared to a non-adaptive application using a single operation mode.

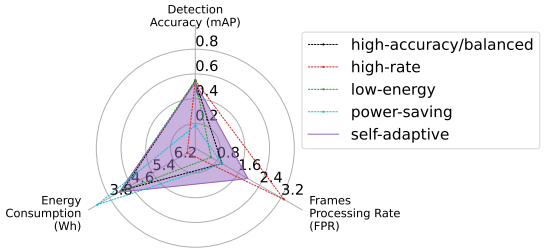
We study this research question in the context of two pedestrian traffic scenarios, namely, *weekdays* and *weekends*, derived from real-world traffic shapes reported by Dobler et al. [46] in their work about urban pedestrians dynamic in the borough of Manhattan. In particular, the *weekdays* scenario has a 3-peaks structure aligned with the "9-to-5" workday time, in which the peaks correspond to commuting to work, exiting buildings at lunch time, and leaving the work place. The *weekend* scenario does not show a peaked structure, but rather a steady increase of pedestrians until the night.

We create a scenario by selecting 1440 frames, that is, 60 frames per hour, from a pool 115 of manually annotated frames containing between 0 and 5 pedestrians. Each hour of the day is labeled as 0 pedestrians, 1 to 3 pedestrians, and 4 to 5 pedestrians. The frames used for the experiment are taken from a study about real-time analytics for traffic safety [9].

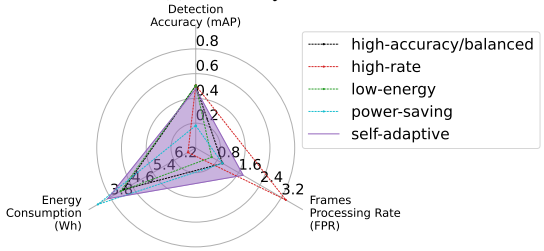
[7]https://docs.voxel51.com/user_guide/evaluation.html

We implement a self-adaptive pedestrian detection application according to the FSM depicted in Fig. 5 using the Python State Machine library (https://pysm.readthedocs.io/). Then, we use the same pedestrian detection logic to obtain the non-adaptive baseline application. The four operation mode configurations obtained by the meta-heuristic search procedure in RQ1 are used to configure both the self-adaptive application and the non-adaptive baselines, obtaining four non-adaptive applications. Fig. 5 shows the configuration parameter values. Further, we include in the study a non-adaptive configuration, namely the *balanced* configuration, that assigns the same weight (0.33) to the three objectives and uses the thresholds ($t_{acc} = 0.3, t_{eng} = 0, t_{rate} = 120$) that filter out the same unsatisfactory configurations collectively filtered out by the four operation modes of the adaptive approach. This configuration implements the best attempt to balance all the objectives without introducing any self-adaptation logic. Interestingly, the *balanced* configuration matches our *high-accuracy* configuration, that is, high-accuracy can be released maintaining a good level of energy consumption and frame rate.

We evaluate the resulting self-adaptive and non-adaptive applications by using the same set of metrics used for RQ1, that is, the MOOP objectives: detection accuracy (mAP), energy consumption (Wh), and FPR.



(a) Weekdays Scenario



(b) Weekends Scenario

Fig. 8: Radar charts comparing the SAA and the 4 non-adaptive applications in the weekdays and weekends scenarios.

Results: Fig. 8 compares the performance of the SAA (purple, solid line) with the four non-adaptive applications (black/red/green/cyan, dotted lines) in both the *weekdays* (Fig. 8a) and *weekends* (Fig. 8b) scenarios. As for the radar charts in Fig. 7, the higher the better.

The shape of the triangle in both the radar charts visually shows how the adaptive behavior guarantees the achievement of a better trade-off among the three objectives compared to the non-adaptive behavior. It outperforms three out of four non-adaptive applications regarding both energy consumption (i.e., *low-energy*, *high-accuracy/balanced*, *high-rate*) and FPR (i.e., *power-saving*, *low-energy*, *high-accuracy/balanced*), and one out of four w.r.t. the detection accuracy (i.e., *power-saving*). Notably, it is still able to guarantee a similar accuracy when compared to the other three non-adaptive applications (i.e., *low-energy*, *high-accuracy/balanced*, *high-rate*).

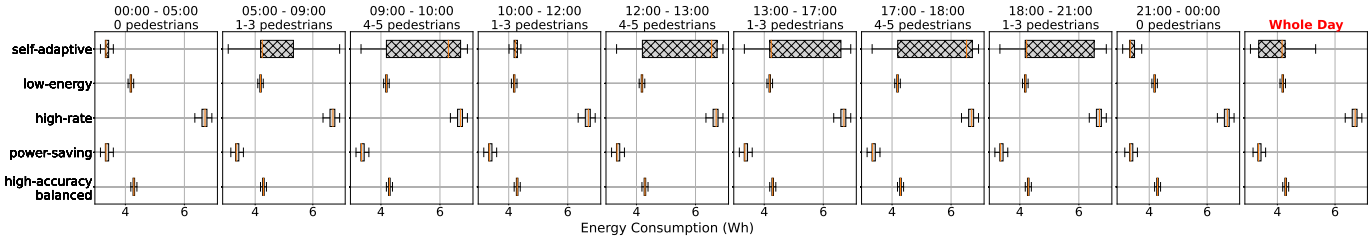
In particular, compared to the best/worst non-adaptive operation mode, the SAA is able to save between 0.5% and 61% of energy in the *weekdays* scenario, and between 13% and 81% in the *weekends* scenario. The improvement on the FPR is between 96% and 233% in the *weekdays* scenario, and between 77% and 196% in the *weekends* scenario. The accuracy loss is between 2% and 4% in the *weekdays* scenario, and between 5% and 6% in the *weekends* scenario, but the SAA outperforms the *power-saving* application with a gain in the accuracy between 62% and 189%.

The SAA performed slightly differently in the two scenarios. In fact, the presence of a 3-peaks structure with a higher number of pedestrians in the *weekdays* scenario makes the self-adaptive application to use more accurate and faster operation modes (i.e., *high-accuracy* and *high-rate*) for a larger amount of time, resulting in a higher FPR at the cost of a higher energy consumption. On the other hand, the traffic shape of the *weekends* scenario fosters the usage of energy efficient operation modes (i.e., *power-saving* and *low-energy*), resulting in a lower energy consumption and slower processing speed. This shows how the SAA application can employ more accurate operation modes when the pedestrians workload is higher, using less accurate operation modes (i.e., *power-saving*) when the pedestrians workload is less demanding.

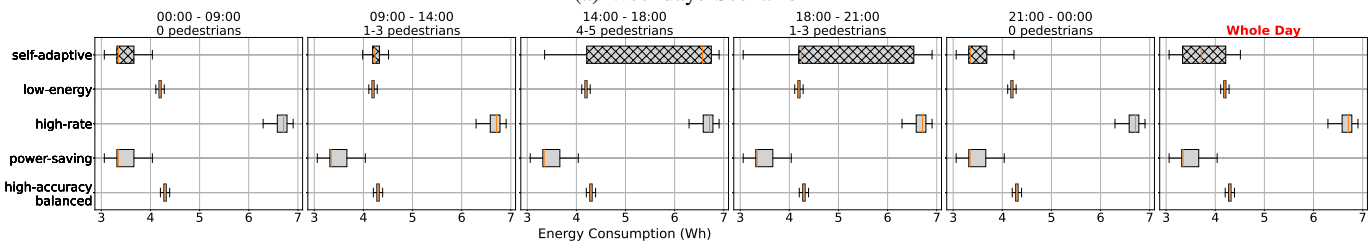
This behavior is also confirmed by the energy consumption box plots shown in Fig. 9a and Fig. 9b. The two figures show the energy consumption of the self-adaptive application and the four non-adaptive applications in different time windows of the day for both the scenarios. The vertical orange line in the boxes indicates the median value.

We can observe how the self-adaptive application captures correctly the 3-peaks structure in the *weekdays* scenario (Fig. 9a) and uses the *high-rate* in these three time windows. At the same time, it employs energy efficient operation modes (i.e., *power-saving* and *low-energy*) when the pedestrians traffic is less intense (e.g., 00:00 - 05:00 and 21:00 - 00:00). A similar behavior is obtained in the *weekends* scenario shown in Fig. 9b. In a nutshell, the self-adaptive solution is consuming energy only when it is worth doing it.

In summary, the empirical evaluation shows how the proposed self-adaptive approach is capable of adapting to a changing environment while balancing multiple application requirements and energy consumption, behaving as optimally as the configurations selected with a near-exhaustive exploration of the parameters space. The experimental material to fully reproduce our study, including instructions to recreate our test-bed based on Raspberry Pi, is avail-



(a) Weekdays Scenario



(b) Weekends Scenario

Fig. 9: Box-plots comparing energy consumption for the self-adaptive and the four non-adaptive applications.

able at https://gitlab.com/sustainable-continuum-monitoring/self-adaptive-moop/-/tree/ASE_2023?ref_type=tags.

D. Threats to Validity

First, the design of the FSM requires the definition of a set of operations modes characterized by weights and thresholds, and the definition of state transition conditions. This is a manual and non-trivial operation guided by domain-expert knowledge that can limit the feasibility of the approach and lead to different results. Nevertheless, the reported results show how a SAA can largely outperform non-adaptive baselines, regardless of the specific configuration used. *Second,* the design of the pedestrian traffic shapes may have an impact on the results. To mitigate this threat, we referred to real scenarios to achieve realistic and informative results. *Finally,* the results may not generalize to other application domains. Indeed, we proposed a *case study* evaluation focusing on AI-services for pedestrian detection running at the edge, and the design of a SAA addressing a different problem may produce different results. Although the methodology and the approach are general, we cannot claim the results shall straightforwardly generalize to other contexts. The illustrated case study nevertheless provides evidence that the proposed approach can generate useful results in non-trivial domains such as pedestrian detection, which requires to balance high-speed computations (e.g., video-processing) with energy saving requirements.

V. Related Work

In the context of IoT architectures and edge oriented systems, self adaptation and optimization technologies have been used to address a range of aspects. For instance, adaptation capabilities have been engineered to achieve *auto-scaling and task offloading* [78], introducing flexibility in the computation at the cost of some jitter in the quality of service and, often, not optimized energy consumption shifts among the nodes [5].

Multiple approaches have been defined to modify the behavior of the components at the edge. The most common examples of self-adaptive edge components are those related to *Adaptive Sampling*. Adaptive sampling refers to the idea of dynamically modifying the sampling rate of sensors and software monitoring probes as well as the inference rate of the components that process such data, according to the context [79]–[82]. Collecting and transmitting less data can save energy and computational resources [83].

Similarly, *Adaptive Filtering* focuses on reducing the number of samples transmitted. For example, if a sensor value is considered similar to a previously collected value or evolves in a predicable way, a node can avoid the transmission of such information to save the transmission cost. Since filtering usually results in sub-optimal performance, the filters must adapt at run-time to guarantee a consistent behavior [79].

Adaptive Compression has been also extensively exploited at the edge. Adaptive Compression solutions aim at reducing the data traffic in the network by reducing the size of the data packets with minimal loss, for instance using strategies that consider the importance of the processed data [84]. Different compression algorithms may also be used dynamically based on the shape of the data, enabling higher compression without inducing significant losses in the accuracy of the data [85].

The approach presented in this paper is complementary to all these forms of adaptation. In fact it provides a methodology to design a SAA running at the edge that is able to adapt its operation mode according to the context. The configurations that correspond to the operation modes are determined empirically, according to the key application objectives that must be optimized. Further optimizing sampling, filtering and compression strategies are additional capabilities.

Self-adaptive behaviors to improve energy consumption have been also studied at the *architectural level* [5]. For instance, a number of approaches have been proposed to

target specific aspects of energy-awareness such as memory handling [86], networking [87], storage [26], and scheduling and provisioning [20]. Furthermore, the ever growing interest in machine learning based solutions lead to specific optimized models for the edge [27]. These solutions can address specific dimensions but lack both the state-based adaptation capabilities introduced in this paper, and the definition of a practical empirical procedure to determine the concrete configurations that must be used by the SAA. Conversely, Da Silva et al. [88] proposed a framework for the automatic generation of application processes. Such processes represent the goals and capabilities of the application in the form of application workflows. This level of adaptation is not usually suitable for edge applications, since the run-time generation of the application processes requires extensive computational capabilities and introduces significant computational overhead [89], which may not be available at edge.

Mobile applications is another domain of self-adaptation where energy consumption is pivotal [90]. While adaptation mechanisms designed for mobile applications are not directly comparable to applications running on the Edge, they share some key aspects, such as the presence of a resource-constrained and battery-powered devices. For instance, Ardito et al. [91], [92] proposed an architectural paradigm in which the operating system or the middleware is able to offer energy-related information to running applications. This enables the implementation of energy-aware self-adaptation strategies based on energy levels. Our proposal is orthogonal with respect to this approach, as we investigate how to design and deploy such applications, with specific focus on those that are AI-based, but without assuming run-time information about the available energy.

VI. Conclusions

We presented an approach that can guide developers in the implementation of AI-based self-adaptive applications able of switching their operation modes in response to changes in the environment. The configuration of the operation modes are determined empirically, based on a meta-heuristic search procedure that can identify useful configurations by sampling a small portion of the configuration space. Experimental results show how the proposed approach can outperform non-adaptive baseline configurations, behaving as optimally as configurations selected with a nearly exhaustive exploration of the configuration space, in a pedestrian detection scenario.

Future work concerns with automating the FSM design and synthesis through data-driven methods, and extending the self-adaptive capabilities by considering clusters of instances that can adapt simultaneously. We also plan to study our approach in a more complex setup involving battery-powered devices and photovoltaic panels, considering run-time energy-related metrics and deploying our prototype in the field.

Acknowledgments

This work has been partially supported by the MUR under the grant "Dipartimenti di Eccellenza 2023-2027", Engineered MachinE Learning-intensive IoT systems (EMELIOT) national research project which has been funded by the MUR under the PRIN 2020 program (Contract 2020W3A5FY), Runtime Control in Multi Clouds (RUCON), Austrian Science Fund (FWF): Y904-N31 START-Programm, 2015, Sustainable Watershed Management Through IoT-Driven Artificial Intelligence (SWAIN), CHIST-ERA-19-CES-005, Austrian Science Fund (FWF), 2021, Standalone Project Transprecise Edge Computing (Triton), Austrian Science Fund (FWF): P 36870-N, 2023, Flagship Project High-Performance Integrated Quantum Computing (HPQC) # 897481 Austrian Research Promotion Agency (FFG), 2023, and by the 5G Use Case Challenge InTraSafEd 5G (Increasing Traffic Safety with Edge and 5G) funded by the City of Vienna.

References

[1] IEA, "Data centres and data transmission networks," IEA, Tech. Rep., 2022.

[2] U. P. POST, "Energy consumption of ICT," UK Parliament, Tech. Rep., 2022.

[3] S. Lange, J. Pohl, and T. Santarius, "Digitalization and energy consumption. does ICT reduce energy demand?" *Ecological Economics*, vol. 176, p. 106760, 2020.

[4] A. Fonseca, R. Kazman, and P. Lago, "A manifesto for energy-aware software," *IEEE Software*, vol. 36, no. 6, pp. 79–82, 2019.

[5] C. Jiang, T. Fan, H. Gao, W. Shi, L. Liu, C. Cérin, and J. Wan, "Energy aware edge computing: A survey," *Computer Communications*, vol. 151, pp. 556–580, 2020.

[6] G. Callebaut, G. Leenders, J. Van Mulders, G. Ottoy, L. De Strycker, and L. Van der Perre, "The art of designing remote iot devices—technologies and strategies for a long battery life," *Sensors*, vol. 21, no. 3, p. 913, 2021.

[7] H. Elahi, K. Munir, M. Eugeni, S. Atek, and P. Gaudenzi, "Energy harvesting towards self-powered iot devices," *Energies*, vol. 13, no. 21, p. 5528, 2020.

[8] V. Pecunia, L. G. Occhipinti, and R. L. Hoye, "Emerging indoor photovoltaic technologies for sustainable internet of things," *Advanced Energy Materials*, vol. 11, no. 29, p. 2100698, 2021.

[9] I. Lujic, V. De Maio, K. Pollhammer, I. Bodrozic, J. Lasic, and I. Brandic, "Increasing traffic safety with real-time edge analytics and 5g," in *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*, 2021, pp. 19–24.

[10] T. S. Combs, L. S. Sandt, M. P. Clamann, and N. C. McDonald, "Automated vehicles and pedestrian safety: exploring the promise and limits of pedestrian detection," *American journal of preventive medicine*, vol. 56, no. 1, pp. 1–7, 2019.

[11] A. M. Nagy and V. Simon, "Survey on traffic prediction in smart cities," *Pervasive and Mobile Computing*, vol. 50, pp. 148–163, 2018.

[12] Z. Lin, H. H. Liu, and M. Wotton, "Kalman filter-based large-scale wildfire monitoring with a system of uavs," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 1, pp. 606–615, 2018.

[13] R. S. Allison, J. M. Johnston, G. Craig, and S. Jennings, "Airborne optical and thermal remote sensing for wildfire detection and monitoring," *Sensors*, vol. 16, no. 8, p. 1310, 2016.

[14] J. P. Dominguez-Morales, L. Duran-Lopez, D. Gutierrez-Galan, A. Rios-Navarro, A. Linares-Barranco, and A. Jimenez-Fernandez, "Wildlife monitoring on the edge: A performance evaluation of embedded neural networks on microcontrollers for animal behavior classification," *Sensors*, vol. 21, no. 9, p. 2975, 2021.

[15] D. Schwartz, J. M. G. Selman, P. Wrege, and A. Paepcke, "Deployment of embedded edge-ai for wildlife monitoring in remote regions," in *2021 20th IEEE International Conference on Machine Learning and Applications*. IEEE, 2021, pp. 1035–1042.

[16] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019.

[17] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Survey and benchmarking of machine learning accelerators," in *2019 IEEE high performance extreme computing conference*. IEEE, 2019, pp. 1–9.

[18] M. Abrar, U. Ajmal, Z. M. Almohaimeed, X. Gui, R. Akram, and R. Masroor, "Energy efficient uav-enabled mobile edge computing for iot devices: A review," *IEEE Access*, vol. 9, pp. 127 779–127 798, 2021.

[19] A. Arouj and A. M. Abdelmoniem, "Towards energy-aware federated learning on battery-powered clients," in *Proceedings of the 1st ACM Workshop on Data Privacy and Federated Learning Technologies for Mobile Edge Network*, 2022, pp. 7–12.

[20] A. Aral, V. De Maio, and I. Brandic, "Ares: Reliable and sustainable edge provisioning for wireless sensor networks," *IEEE Transactions on Sustainable Computing*, 2021.

[21] A. F. Aljulayfi and K. Djemame, "A novel qos and energy-aware self-adaptive system architecture for efficient resource management in an edge computing environment," in *35th UK Performance Engineering Workshop 16 December 2019*, 2019, p. 39.

[22] R. Ghosh, S. P. R. Komma, and Y. Simmhan, "Adaptive energy-aware scheduling of dynamic event analytics across edge and cloud resources," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2018, pp. 72–82.

[23] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, Y. Dou, and A. Y. Zomaya, "Adaptive energy-aware computation offloading for cloud of things systems," *IEEE access*, vol. 5, pp. 23 947–23 957, 2017.

[24] J. Silva, E. R. Marques, L. Lopes, and F. Silva, "Energy-aware adaptive offloading of soft real-time jobs in mobile edge clouds," *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–21, 2021.

[25] I. U. Khan, M. A. Hassan, M. D. Alshehri, M. A. Ikram, H. J. Alyamani, R. Alturki, and V. T. Hoang, "Monitoring system-based flying iot in public health and sports using ant-enabled energy-aware routing," *Journal of Healthcare Engineering*, vol. 2021, 2021.

[26] R. Vales, J. Moura, and R. Marinheiro, "Energy-aware and adaptive fog storage mechanism with data replication ruled by spatio-temporal content popularity," *Journal of Network and Computer Applications*, vol. 135, pp. 84–96, 2019.

[27] I. Brandic, "Sustainable and trustworthy edge machine learning," *IEEE Internet Computing*, vol. 25, no. 5, pp. 5–9, 2021.

[28] H. Sahar, A. A. Bangash, and M. O. Beg, "Towards energy aware object-oriented development of android applications," *Sustainable Computing: Informatics and Systems*, vol. 21, pp. 28–46, 2019.

[29] A. E. Trefethen and J. Thiyagalingam, "Energy-aware software: Challenges, opportunities and strategies," *Journal of Computational Science*, vol. 4, no. 6, pp. 444–449, 2013.

[30] K. Eder, J. P. Gallagher, G. Fagas, L. Gammaitoni, and D. Paul, "Energy-aware software engineering," *ICT-energy concepts for energy efficiency and sustainability*, pp. 103–127, 2017.

[31] K. Chinnappan, I. Malavolta, G. A. Lewis, M. Albonico, and P. Lago, "Architectural tactics for energy-aware robotics software: A preliminary study," in *Software Architecture: 15th European Conference, ECSA 2021, Virtual Event, Sweden, September 13-17, 2021, Proceedings*. Springer, 2021, pp. 164–171.

[32] S. Vos, P. Lago, R. Verdecchia, and I. Heitlager, "Architectural tactics to optimize software for energy efficiency in the public cloud," in *2022 International Conference on ICT for Sustainability*. IEEE, 2022, pp. 77–87.

[33] J. Panerati, D. Sciuto, and G. Beltrame, "Optimization strategies in design space exploration," in *Handbook of Hardware/Software Codesign*. Springer, 2017, pp. 189–216.

[34] G. H. S. Association. (2023) Pedestrian traffic fatalities by state: 2022 preliminary data. Accessed on 2023-04-12. [Online]. Available: https://www.ghsa.org/resources/Pedestrians23

[35] T. Mohn. (2023) Pedestrian deaths on the rise again, a walker dies every 75 minutes on america's roads. Accessed on 2023-04-12. [Online]. Available: https://www.forbes.com/sites/tanyamohn/2023/02/28/pedestrian-deaths-on-the-rise--again-a-walker-dies-every-75-minutes-on-americas-roads/

[36] E. T. S. Council. (2023) Five ways europe can tackle road deaths. Accessed on 2023-04-12. [Online]. Available: https://etsc.eu/five-ways-europe-can-tackle-road-deaths/

[37] C. S. Service. (2023) Road safety in the eu. Accessed on 2023-04-12. [Online]. Available: https://ec.europa.eu/commission/presscorner/detail/en/ip_23_953

[38] Centers for Disease Control and Prevention. (2022) Pedestrian safety. Accessed on 2023-03-09. [Online]. Available: https://www.cdc.gov/transportationsafety/pedestrian_safety/index.html

[39] R. Shreyas, B. P. Kumar, H. Adithya, B. Padmaja, and M. Sunil, "Dynamic traffic rule violation monitoring system using automatic number plate recognition with sms feedback," in *2017 2nd International Conference on Telecommunication and Networks*. IEEE, 2017, pp. 1–5.

[40] S. Alagarsamy, S. Ramkumar, K. Kamatchi, H. Shankar, A. Kumar, S. Karthick, and P. Kumar, "Designing a advanced technique for detection and violation of traffic control system," *Journal of Critical Reviews*, vol. 7, no. 8, pp. 2874–2879, 2020.

[41] M. Bolsunovskaya, A. Leksashov, S. Shirokova, and V. Tsygan, "Development of an information system structure for photo-video recording of traffic violations," in *E3S Web of Conferences*, vol. 244. EDP Sciences, 2021, p. 07007.

[42] S. Amini, I. Gerostathopoulos, and C. Prehofer, "Big data analytics architecture for real-time traffic control," in *2017 5th IEEE international conference on models and technologies for intelligent transportation systems*. IEEE, 2017, pp. 710–715.

[43] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors*, vol. 19, no. 9, p. 2048, 2019.

[44] G.-D. Voinea, C. C. Postelnicu, M. Duguleana, G.-L. Mogan, and R. Socianu, "Driving performance and technology acceptance evaluation in real traffic of a smartphone-based driver assistance system," *International journal of environmental research and public health*, vol. 17, no. 19, p. 7098, 2020.

[45] I. Lashkov and A. Kashevnik, "Smartphone-based intelligent driver assistant: context model and dangerous state recognition scheme," in *Intelligent Systems and Applications: Proceedings of the 2019 Intelligent Systems Conference Volume 2*. Springer, 2020, pp. 152–165.

[46] G. Dobler, J. Vani, and T. T. L. Dam, "Patterns of urban foot traffic dynamics," *Computers, Environment and Urban Systems*, vol. 89, p. 101674, 2021.

[47] Y. Brun, G. Di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, "Engineering self-adaptive systems through feedback loops," *Software engineering for self-adaptive systems*, pp. 48–70, 2009.

[48] M. Salehie and L. Tahvildari, "Towards a goal-driven approach to action selection in self-adaptive software," *Software: Practice and Experience*, vol. 42, no. 2, pp. 211–233, 2012.

[49] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive and Mobile Computing*, vol. 17, pp. 184–206, 2015.

[50] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[51] C. Ghezzi, A. Mocci, and M. Sangiorgio, "Runtime monitoring of component changes with spy@ runtime," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 1403–1406.

[52] G. Karsai and J. Sztipanovits, "A model-based approach to self-adaptive software," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 3, pp. 46–53, 1999.

[53] P. Arcaini, E. Riccobene, and P. Scandurra, "Formal design and verification of self-adaptive systems with decentralized control," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 11, no. 4, pp. 1–35, 2017.

[54] E. Lee, Y.-D. Seo, and Y.-G. Kim, "Self-adaptive framework based on mape loop for internet of things," *sensors*, vol. 19, no. 13, p. 2996, 2019.

[55] L. Aultman-Hall, D. Lane, and R. R. Lambert, "Assessing impact of weather and season on pedestrian traffic volumes," *Transportation research record*, vol. 2140, no. 1, pp. 35–43, 2009.

[56] T. Kim, D.-W. Sohn, and S. Choo, "An analysis of the relationship between pedestrian traffic volumes and built environment around metro stations in seoul," *KSCE Journal of Civil Engineering*, vol. 21, pp. 1443–1452, 2017.

[57] T. Rupprecht and Y. Wang, "A survey for deep reinforcement learning in markovian cyber-physical systems: Common problems and solutions," *Neural Networks*, 2022.

[58] W. Stadler, *Multicriteria Optimization in Engineering and in the Sciences*. Springer Science & Business Media, 1988, vol. 37.

[59] P. Ngatchou, A. Zarei, and A. El-Sharkawi, "Pareto multi objective optimization," in *Proceedings of the international conference on, intelligent systems application to power systems*. IEEE, 2005, pp. 84–91.

[60] S. Verma, M. Pant, and V. Snasel, "A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems," *Ieee Access*, vol. 9, pp. 57 757–57 791, 2021.

[61] 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, 2002.

[62] Y. Kuo, T. Yang, and G.-W. Huang, "The use of grey relational analysis in solving multiple attribute decision-making problems," *Computers & industrial engineering*, vol. 55, no. 1, pp. 80–93, 2008.

[63] Z. Wang and G. P. Rangaiah, "Application and analysis of methods for selecting an optimal solution from the pareto-optimal front obtained by multiobjective optimization," *Industrial & Engineering Chemistry Research*, vol. 56, no. 2, pp. 560–574, 2017.

[64] D. Ju-Long, "Control problems of grey systems," *Systems & control letters*, vol. 1, no. 5, pp. 288–294, 1982.

[65] G. Chia Yee, C. Jeng Feng, M. A. B. Chik, and M. Mokhtar, "Weighted grey relational analysis to evaluate multilevel dispatching rules in wafer fabrication," *Grey Systems: Theory and Application*, vol. 11, no. 4, pp. 619–649, 2021.

[66] A. Mahmoudi, S. A. Javed, S. Liu, and X. Deng, "Distinguishing coefficient driven sensitivity analysis of gra model for intelligent decisions: application in project management," *Technological and Economic Development of Economy*, vol. 26, no. 3, pp. 621–641, 2020.

[67] H.-H. Wu, "A comparative study of using grey relational analysis in multiple attribute decision making problems," *Quality Engineering*, vol. 15, no. 2, pp. 209–217, 2002.

[68] C.-J. Hsu and C.-Y. Huang, "Comparison of weighted grey relational analysis for software effort estimation," *Software Quality Journal*, vol. 19, pp. 165–200, 2011.

[69] C. Pruteanu and C.-g. Haba, "Genfsm: A finite state machine generation tool," in *Proc. 9th Int. Conf. Dev. Applicat. Syst*, 2008, pp. 165–168.

[70] B. Vandeportaele, "A finite state machine modeling language and the associated tools allowing fast prototyping for fpga devices," in *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics*. IEEE, 2017, pp. 1–6.

[71] P. Adamczyk, "Selected patterns for implementing finite state machines," in *The 11th Conference on Pattern Languages of Programs*, 2004.

[72] ——, "The anthology of the finite state machine design patterns," in *The 10th Conference on Pattern Languages of Programs*, 2003.

[73] S. M. Yacoub and H. H. Ammar, *Pattern-oriented analysis and design: composing patterns to design software systems*. Addison-Wesley Professional, 2004.

[74] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[75] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[76] P. Dendorfer, A. Osep, A. Milan, K. Schindler, D. Cremers, I. Reid, S. Roth, and L. Leal-Taixé, "Motchallenge: A benchmark for single-camera multiple target tracking," *International Journal of Computer Vision*, vol. 129, pp. 845–881, 2021.

[77] R. Padilla, S. L. Netto, and E. A. Da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 international conference on systems, signals and image processing*. IEEE, 2020, pp. 237–242.

[78] I. Alfonso, K. Garcés, H. Castro, and J. Cabot, "Modeling self-adaptative iot architectures," in *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion*. IEEE, 2021, pp. 761–766.

[79] D. Giouroukis, A. Dadiani, J. Traub, S. Zeuch, and V. Markl, "A survey of adaptive sampling and filtering algorithms for the internet of things," in *Proceedings of the 14th ACM International Conference on Distributed and Event-based Systems*, 2020, pp. 27–38.

[80] V. Colombo, A. Tundo, M. Ciavotta, and L. Mariani, "Towards self-adaptive peer-to-peer monitoring for fog environments," in *Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2022, pp. 156–166.

[81] J. Mertz and I. Nunes, "Software runtime monitoring with adaptive sampling rate to collect representative samples of execution traces," *Journal of Systems and Software*, p. 111708, 2023.

[82] D. Yuan, S. Park, and Y. Zhou, "Characterizing logging practices in open-source software," in *2012 34th International Conference on Software Engineering*. IEEE, 2012, pp. 102–112.

[83] A. Jain and E. Y. Chang, "Adaptive sampling for sensor networks," in *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, 2004, pp. 10–16.

[84] T. Lu, W. Xia, X. Zou, and Q. Xia, "Adaptively compressing iot data on the resource-constrained edge," in *3rd {USENIX} Workshop on Hot Topics in Edge Computing*, 2020.

[85] J. Cheng, Q. Ye, H. Jiang, D. Wang, and C. Wang, "Stcdg: An efficient data gathering algorithm based on matrix completion for wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 2, pp. 850–861, 2012.

[86] G. Jia, G. Han, J. Du, and S. Chan, "A maximum cache value policy in hybrid memory-based edge computing for mobile devices," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4401–4410, 2018.

[87] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[88] C. E. da Silva and R. de Lemos, "A framework for automatic generation of processes for self-adaptive software systems," *Informatica*, vol. 35, no. 1, 2011.

[89] J. Cámara, D. Garlan, B. Schmerl, and A. Pandey, "Optimal planning for architecture-based self-adaptation via model checking of stochastic games," in *Proceedings of the 30th annual ACM symposium on applied computing*, 2015, pp. 428–435.

[90] E. M. Grua, I. Malavolta, and P. Lago, "Self-adaptation in mobile apps: a systematic literature study," in *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2019, pp. 51–62.

[91] L. Ardito, "Energy aware self-adaptation in mobile systems," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 1435–1437.

[92] L. Ardito, M. Torchiano, M. Marengo, and P. Falcarin, "glcb: an energy aware context broker," *Sustainable Computing: Informatics and Systems*, vol. 3, no. 1, pp. 18–26, 2013.