ARES: Reliable and Sustainable Edge Provisioning for Wireless Sensor Networks

Atakan Aral, *Member, IEEE,* Vincenzo De Maio, *Member, IEEE,* and Ivona Brandic, *Member, IEEE*

Abstract—Wireless sensor networks have wide applications in monitoring applications. However, sensors' energy and processing power constraints, as well as the limited network bandwidth, constitute significant obstacles to near-real-time requirements of modern IoT applications. Offloading sensor data on an edge computing infrastructure instead of in-cloud or in-network processing is a promising solution to these issues. Nevertheless, due to (1) geographical dispersion, (2) ad-hoc deployment, and (3) rudimentary support systems compared to cloud data centers, reliability is a critical issue. This forces edge service providers to deploy a huge amount of edge nodes over an urban area, with catastrophic effects on environmental sustainability. In this work, we propose ARES, a two-stage optimization algorithm for sustainable and reliable deployment of edge nodes in an urban area. Initially, ARES applies multi-objective optimization to identify a set of Pareto-optimal solutions for transmission time and energy; then it augments these candidates in the second stage to identify a solution that guarantees the desired level of reliability using a dynamic Bayesian network based reliability model. ARES is evaluated through simulations using data from the urban area of Vienna. Results demonstrate that it can achieve a better trade-off between transmission time, energy-efficiency, and reliability than the state-of-the-art solutions.

Index Terms—Edge Computing, Wireless Sensor Networks, Provisioning, Energy-Efficiency, Latency, Fault-Tolerance, QoS

1 Introduction

Advancements in microelectromechanical technology have enabled mass production of various inexpensive sensors, enhanced with limited data processing and transmission capabilities. These so-called smart sensors have been widely adopted to monitor and record physical conditions in many areas but particularly in automotive, healthcare, and industrial automation. The global smart sensor market was valued at USD 36.62 billion in 2019 and estimated to triple by 2025 [1]. Sensors are typically battery-powered and deployed in an ad-hoc and spatially dispersed manner as wireless sensor networks (WSN). Due to their limited network, energy, and computational resources [2], WSN relies on wireless connectivity to offload data processing.

However, the energy consumption for communication is a serious issue in this scenario. It is estimated that the transmission of a single bit of data requires the same amount of energy as executing 50 to 150 instructions [3]. Moreover, near real-time applications may suffer from intolerable delays due to the long-distance communication with remote servers. The global average round-trip time from the edge of the network to a remote cloud data center is estimated as 74 ms [4]. Delays in communications are especially critical in applications like `InTraSafEd5G`[1], developed by our research group at Vienna University of Technology. It is a traffic-safety application supporting drivers by signaling critical situations in their blind spots. `InTraSafEd5G` requires the processing of heterogeneous data coming from different sensors to identify critical situations and deliver notifications to drivers to allow timely reactions.

- *A. Aral, V. De Maio, and I. Brandic are with the Institute of Information Systems Engineering, Vienna University of Technology, Vienna, Austria. E-mail: {atakan,vincenzo,ivona}@ec.tuwien.ac.at*
- *A. Aral and V. De Maio contributed equally to this work as first authors.*

1. https://newsroom.magenta.at/2020/01/16/5g-anwendungen/

Alternative solutions to WSN data offloading such as in-network processing techniques [5] advocate utilizing idle resources of sensor nodes for processing sensor data. Even though in-network data processing consumes substantially less energy than communication [6], this approach is not suitable for complex data analytics tasks (i.e., deep learning) due to high resource needs. In-network processing is also prone to failures, which is an additional source of delays.

We propose processing WSN data on an edge computing infrastructure as a solution to these issues. Edge computing performs data processing on computational nodes placed in close proximity to data sources (e.g., sensors) [7]. The adoption of this paradigm allows combining the benefits of both centralized and in-network processing. Nevertheless, to exploit the distinctive features of edge computing in this context, it is of paramount importance to provision edge nodes (ENs) ensuring (1) low transmission time and high reliability to address near real-time requirements; (2) low energy consumption of sensors and ENs to ensure environmental sustainability and maximize the sensors' lifetime.

Designing a provisioning method to find a trade-off solution for data transmission, sustainability and reliability poses several challenges due to the complex relationships between these three objectives. For example, active-standby or load sharing replication is considered the most effective fault-tolerance technique for latency-sensitive edge applications [8]; however, redundancy would result in higher energy consumption. On the other hand, provisioning a particular set of most central ENs to minimize communication distance could expose them to overloading and higher failure risk. To address these challenges, we propose ARES (sustAinable and Resilient Edge proviSioning), an offline algorithm for the sustainable and resilient placement of ENs based on the geographical distribution of sensors as well as energy and failure characteristics of available resources.

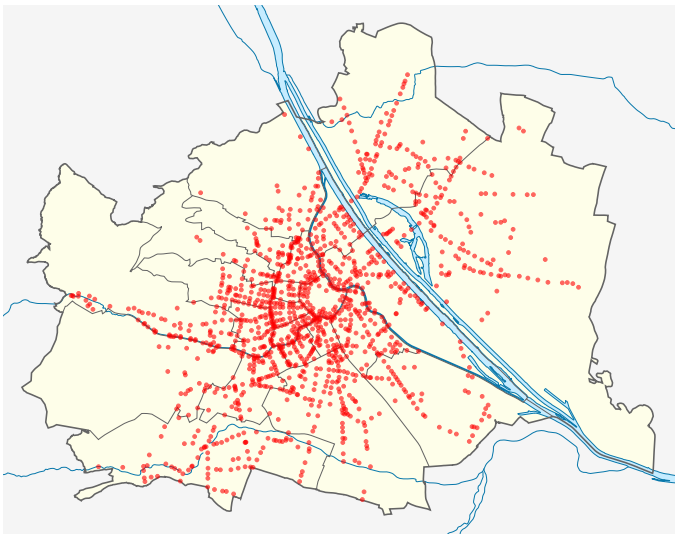


Fig. 1. Prospective Distribution of Smart Traffic Lights in Vienna [13].

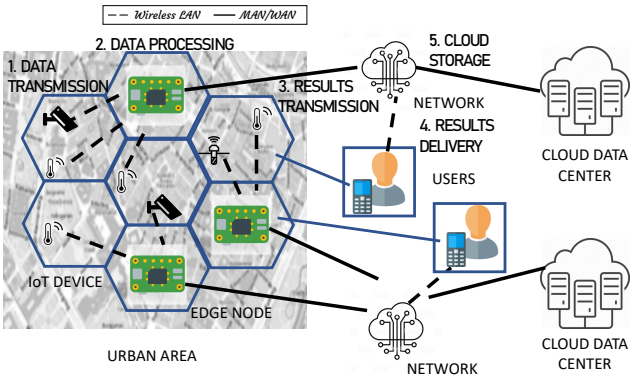


Fig. 2. Proposed WSN-Edge Offloading Architecture.

The main novelty of this work lies in the implication of correlated failures in the sustainable edge provisioning. To the best of our knowledge, previous work either does not consider reliability at all or assumes independent failures, which is an oversimplification according to our results as well as prior analyses. Our findings would be precious for many stakeholders of IoT systems, including telecommunications and telemetry providers for cost-efficient capacity planning. We demonstrate the benefits of our approach using `InTraSafEd5G` project as a concrete example.

This article is organized as follows. First, we provide background information on WSN in Section 2 along with a motivational use case. In Section 3 we provide an overview of ARES, then in Section 4, we describe our theoretical system model. In Section 5, we explain both stages of the ARES method in detail along with worst-case performance analysis. We describe the experimental setup and discuss the numerical evaluation results in Sections 6 and 7, respectively. Finally, we survey the literature in Section 8 and conclude the article with future directions in Section 9.

2 Background

WSN refers to spatially dispersed sensors for monitoring and recording environmental conditions. Sensor data are then transmitted to different locations to be processed. The measurement parameters typical of WSNs are temperature, sound, pollution levels, humidity, wind, etc. Such sensors usually are connected through ad-hoc networks [9]. WSNs are used in many scenarios including e-health [10], natural disaster prevention [11], and water pollution monitoring [12]. However, centralized data analytics can be energy-inefficient for WSNs. As an alternative, in-network data processing, i.e., distributed and collaborative data processing performed by the WSN nodes has been proposed [5], [6].

Two of the most widely adopted uses of WSNs in urban environments are air pollution and temperature monitoring. They have been deployed in several metropolitan cities such as London, Stockholm, and Vienna [14]. Some examples of uses are the detection of malfunctioning air quality filters in industries and the sustainable operation of district cooling systems based on detected urban heat islands. Recently, the *Smart City Wien* initiative by the city of Vienna has announced that all Viennese traffic signal systems (Figure 1) are being equipped with a total of approximately 10,000 weather and environmental sensors[2]. Such complex systems face the following challenges: (1) coverage of a geographically wide area; (2) continuous generation of a large amount of sensor data; (3) near-real-time processing of streaming data. These challenges cannot be solved by typical cloud-based WSN architectures, due to the high latency required to transfer data to cloud data centers for aggregation and processing. Therefore, we enhance typical WSN data processing by employing hybrid cloud/edge infrastructures. Among various prospective deployment strategies for edge computing, we consider the devices at the extreme edge of the network infrastructure to achieve the lowest latency.

Deploying such hybrid cloud/edge infrastructures requires the placement of ENs in the proximity of sensors to reduce the data transmission time. We envision the scenario in Figure 2. Data coming from different sensors are first collected and transferred to computational nodes (Step 1), where they are processed (Step 2). Processing output is transferred over the network (Step 3) and delivered to users (Step 4). If long-term large-scale analytics are needed, a summary of the data can be eventually stored in the cloud (Step 5). Most of the applications relying on WSN have strict near real-time requirements. Since such requirements are typical of critical systems, also a high degree of reliability has to be ensured. However, placement of a great number of ENs over an urban area raises a sustainability issue, due to the additional energy consumption required by ENs.

3 ARES Overview

In this work, we design ARES, a two-stage optimization method for *sustainable* placement of ENs ensuring *low transmission time* and *high reliability*, which are of paramount importance in the WSN scenario [2]. The rationale behind the choice of two-stage optimization lies in the fact that, while transmission time and energy consumption can be estimated relatively fast on a per-node basis, we need to consider the fault-tolerance of the whole deployment due

2. https://smartcity.wien.gv.at/site/en/smart-traffic-lights/

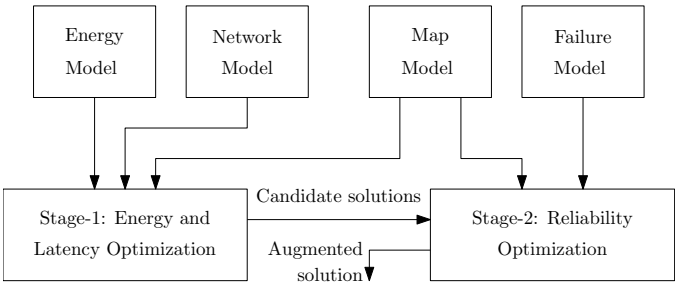


Fig. 3. A high-level overview of major ARES components.

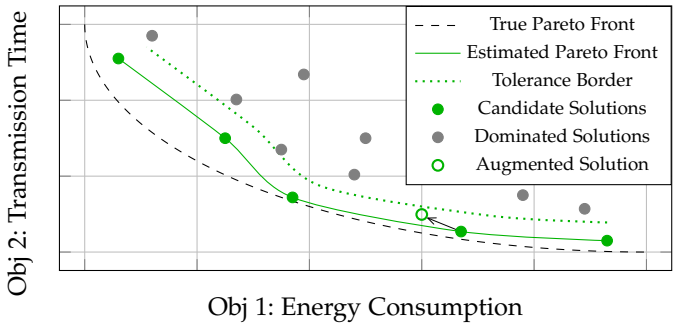


Fig. 4. Illustration of ARES method with candidate solutions from stage-1 and their augmentation towards a reliable solution at stage-2.

to the correlated failures [15] to compute reliability. In the context of edge computing, the causes of spatio-temporal correlation include: network failures affecting nodes in the same physical/virtual network; power outages affecting nodes in the same power grid; nodes deployed in hostile locations failing due to environmental/weather interference; and cascading failures caused by overloading [8]. These factors greatly complicate the global reliability optimization. Accordingly, we first reduce the search space by excluding provisionings that are inefficient in terms of transmission time and energy consumption (i.e., dominated solutions). Then, reliability is handled as a local optimization of only the trade-off solutions for these two objectives.

A high-level overview of the ARES components and the data flows among them are illustrated in Figure 3. In the first stage, Pareto-optimal *candidate solutions* for transmission time and energy consumption are identified; in the second stage, these candidate solutions are evaluated in terms of reliability and modified to obtain a single *augmented solution*. We choose to work on a set of Pareto-optimal solutions rather than a single optimal solution because of the significantly higher time required to compute the latter and to allow a wider exploration of the solution space. Deviations from the Pareto-optimality within a tolerance interval are allowed in the second stage as shown in Figure 4.

We consider the WSN-edge offloading scenario in Figure 2 and assume that the urban area is divided into hexagonal cells, as typical in mobile cellular networks [16]. The sensors are connected through ad-hoc wireless networks rather than wired installations to simplify the wide-area deployment as shown in Figure 1. Workload distribution (i.e., locations of sensors and output rates) is expected to be relatively stable in the considered use cases; therefore, we deal with offline provisioning. ENs are considered small clusters of single-board computers like Raspberry PI [17].

4 Theoretical Model

The edge provisioning problem targeted in this work needs evaluation of the three objectives before ENs can be deployed and hence the actual values can be measured. Therefore, ARES requires their realistic estimation to make accurate provisioning decisions. In this section, we describe the theoretical foundations of our work, where we meticulously model the network infrastructure for data transmission, urban area map, energy consumption, and failures. These models are then employed by both stages of ARES in estimating the optimization objectives as shown in Figure 4. Notations used in this section are summarized in Table 1.

4.1 Network Infrastructure Model

We define network as an undirected graph $\mathcal{I} \stackrel{\text{def}}{=} (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ the set of network connections. $\mathcal{N}$ is defined as $\mathcal{N} \stackrel{\text{def}}{=} \mathcal{C} \cup \mathcal{S}$, where $\mathcal{C}$ is the set of *compute* nodes and $\mathcal{S}$ the set of *sensor* nodes. The set $\mathcal{C}$ is defined as $\mathcal{C} \stackrel{\text{def}}{=} \mathcal{C}_c \cup \mathcal{C}_e$, respectively the set of cloud and edge nodes, for which we define also computational capabilities (number of CPUs, millions of instructions per second (MIPS)). We assume that sensor nodes have no computational capabilities. We define $\mathcal{L}$ as a subset of $\mathcal{N} \times \mathcal{N}$. For each $(n_i, n_j) \in \mathcal{L}$ we define $latency(n_i, n_j, \tau)$ and $bw(n_i, n_j, \tau)$ respectively as latency bandwidth available between n_i and n_j at instant τ. Latency and bandwidth available at instant τ are modeled by random variables, whose distribution is based on real traces collected in [18].

4.2 Map Model

The urban area map is modeled as a grid where network nodes are deployed. The area is divided into hexagonal cells, which identify clusters of nodes managed by a single node. This is typical in mobile cellular networks [16]. The map $\mathcal{M}_{m,n}$ is defined as a $m \times n, m, n \in \mathbf{N}$ hexagonal grid. We employ a doubled coordinates system to identify each cell with X coordinates defined as $X = \{x \in \mathbb{N} : x < 2m\}$, and Y coordinates as $Y = \{y \in \mathbb{N} : y < n\}$ as in Figure 5. A cell neighborhood is defined in Equation 1, whereas neighbors of distance d are defined recursively in Equations 2 and 3.

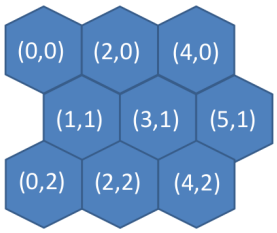


Fig. 5. 3×3 hexagonal grid with doubled coordinates.

$$neigh(x,y) \stackrel{\text{def}}{=} \{(x',y') : (x',y') = (x+1, y+1), (x+1, y-1), (x, y-2), (x-1, x-1), (x-1, y+1), (x, y+2) : x', y' \geq 0\}. \quad (1)$$

$$neigh(x,y,0) = \{(x,y)\}. \quad (2)$$

TABLE 1
Summary of notation used in this article.

Symbol	Description
$\mathcal{T}(\mathcal{P})$	Transmission time for provisioning $\mathcal{P}$
$\mathcal{E}(\mathcal{P})$	Energy consumption for provisioning $\mathcal{P}$
$\mathcal{R}(\mathcal{P})$	Reliability for provisioning $\mathcal{P}$
$\mathcal{L}_e$	Admissible locations for edge nodes
$\mathcal{P}$	Provisioning of infrastructure
$\mathcal{F}$	Set of non-dominated provisioning
$\mathcal{S}$	Set of sensors in the infrastructure
$\mathcal{C}_c$	Set of cloud nodes in the infrastructure
$\mathcal{C}_e$	Set of ENs in the infrastructure
$\mathcal{I}$	Cloud/edge infrastructure to provision
$\mathcal{L}$	Set of network connections between nodes
$\mathcal{M}_{m,n}$	Map of the urban area
$m(i,j)$	Cell i,j in map $\mathcal{M}_{m,n}$
$\mathcal{C}_e^{\mathcal{P}}$	ENs provisioned in $\mathcal{P}$
$\mathcal{D}_e$	Placement of each EN on the map
$\mathcal{D}_e(k)$	Cell $m(i,j)$ where EN e_k is placed
$\mathcal{M}_{m,n}(i,j)$	Set of ENs deployed in cell $m(i,j)$ of map $\mathcal{M}_{m,n}$
$P_n(n,\tau)$	Istantaneous power on node n at time τ
$P_{cpu}(n,\tau)$	CPU power consumption on node n at time τ
$\mathcal{U}_{cpu}(n,\tau)$	Utilization of CPU on node n at time τ
$\mathcal{U}_{net}(n,\tau)$	Utilization of network on node n at time τ
$thr(n)$	Load level on node n where trend of P_{cpu} changes
$P^l_{cpu}(n,\tau)$	Function modelling CPU power consumption when $\mathcal{U}_{cpu}(n,\tau) \leq thr(n)$
$P^h_{cpu}(n,\tau)$	Function modelling CPU power consumption when $\mathcal{U}_{cpu}(n,\tau) > thr(n)$
$P_{net}(n,\tau)$	NET power consumption on node n at time τ
$P_{idle}(n)$	Idle power on node n
$P_{active}(n,\tau)$	Active power on node n at time τ
$\alpha(n),\beta(n),\gamma(n)$	Coefficients used for power functions on node n
f_k	Random binary event for the failure of node k
$f_{\mathcal{P}}$	Random binary event for the failure of $\mathcal{P}$

$$\forall d \in \mathbb{N}^0 \left(neigh(x,y,d+1) = \bigcup_{\substack{(x',y')\in \\ neigh(x,y,d)}} neigh(x',y') \right). \quad (3)$$

The distance between cells is defined by Equation 4.

$$dist((x_1,y_1),(x_2,y_2)) = |x_1 - x_2| + \max(0, \frac{|x_1-x_2| - |y_1-y_2|}{2}). \quad (4)$$

While cloud nodes are deployed outside of the urban area, ENs and sensors in $\mathcal{I}$ are deployed over $\mathcal{M}_{m,n}$. In this work, we focus on the provisioning of ENs. We assume that ENs can be deployed only in specific locations, as typical in many big cities due to urbanistic and space requirements. We define the set of admissible locations for edge nodes as $\mathcal{L}_e$, where each location $l \in \mathcal{L}_e$ corresponds to GPS coordinates in the map $\mathcal{M}_{m,n}$. Multiple $l \in \mathcal{L}_e$ can belong to the same $m(i,j)$ cell. Then, we define the set of provisioned ENs as $\mathcal{C}_e^{\mathcal{P}}$. For each EN e_k in $\mathcal{C}_e^{\mathcal{P}}$ we define the GPS coordinates as $coords(e_k)$. The deployment of ENs is defined by a vector $\mathcal{D}_e$ of size $|\mathcal{C}_e^{\mathcal{P}}|$, where $\mathcal{D}_e(e_k)$ contains the cell $m(i,j) \in \mathcal{M}_{m,n}$ where the node e_k is deployed. A deployment of ENs is *admissible* only if each EN is deployed in admissible coordinates, namely,

$$\mathcal{D}^e \text{ is admissible } \iff \forall e_k \in \mathcal{C}_e^{\mathcal{P}}, coords(e_k) \in \mathcal{L}_e.$$

For simplicity, we define $\mathcal{M}_{m,n}(i,j)$ as the set of ENs deployed in cell $m(i,j)$, namely, $\mathcal{M}_{m,n}(i,j) \stackrel{\text{def}}{=} \{e_k \in \mathcal{C}_e^{\mathcal{P}} : \mathcal{D}_e(e_k) = m(i,j)\}$. Finally, we define a provisioning for $\mathcal{I}$ over $\mathcal{M}_{m,n}$ as the quadruple $\mathcal{P} \stackrel{\text{def}}{=} \langle \mathcal{I}, \mathcal{M}_{m,n}, \mathcal{C}_e^{\mathcal{P}}, \mathcal{D}_e \rangle$.

4.3 Transmission Time

Transmission time is defined as the time required to transfer data from a sensor $s \in \mathcal{S}$ to a computational node $n \in \mathcal{C}$. We define the transmission time between s and n as,

$$t(s,n) = latency(s,n,\tau) \cdot dist(s,n) + \frac{data(s)}{bw(s,n,\tau)}. \quad (5)$$

where $data(s)$ is the amount of data transferred by sensor s. Let $\tilde{T}(s)$ be the time required to transmit from a sensor s to its closest computational node, namely,

$$\tilde{T}(s) = \min_{n \in \mathcal{C}_e} t(s,n). \quad (6)$$

The goal of ARES is to find a provisioning that minimizes the maximum $\tilde{T}(s)$ for each $s \in \mathcal{I}$, namely $\mathcal{T}(\mathcal{P}) = \min \max_{s \in \mathcal{S}} \tilde{T}(s)$. In this work, we do not consider network failures, as we address offline provisioning. However, since this model is based on the size of data transfer and available latency and bandwidth, it can be applied also in the case of data retransmission.

4.4 Energy Consumption

The energy model used in this work is adapted from [19]. Energy consumption of provisioning $\mathcal{E}(\mathcal{P})$ is defined as the integral of the instantaneous power of computational nodes P_n over time τ. As in [19], instantaneous power consumption of a computational node is composed of an idle part, $P_{idle}(n)$, and an active part, P_{active}. Therefore, the power consumption of a single $n \in \mathcal{N}$ is defined as

$$P_n(n,\tau) = P_{idle}(n) + P_{active}(n,\tau), \quad (7)$$

where P_{active} is the sum of the $P_{cpu}(n,\tau)$ and $P_{net}(n,\tau)$ and $P_{idle}(n)$ is a hardware-defined constant dependent on node n. We define power consumption of a computational node as a piecewise linear function, i.e. using two different functions for different levels of load on node n, as in [19]. Let $\mathcal{U}_{cpu}(n,\tau)$ be the instantaneous CPU utilization on node n at instant τ and $thr(n)$ the load level at which the tendency of $P_{cpu}(n,\tau)$ changes. The instantaneous power function is defined in Equation 8.

$$P_{cpu}(n,\tau) \stackrel{\text{def}}{=} \begin{cases} P^l_{cpu}(n,\tau), & \text{if } \mathcal{U}_{cpu}(n,\tau) \leq thr(n); \\ P^h_{cpu}(n,\tau), & \text{otherwise.} \end{cases} \quad (8)$$

$P^l_{cpu}(n,\tau)$ and $P^h_{cpu}(n,\tau)$ are defined in Equations 9 and 10.

$$P^l_{cpu}(n,\tau) \stackrel{\text{def}}{=} \alpha(n) \cdot P_r(n) \cdot \mathcal{U}_{cpu}(n,\tau) \quad (9)$$

$$P^h_{cpu}(n,\tau) \stackrel{\text{def}}{=} \beta(n) \cdot P_r(n) + (1-\beta(n)) \cdot P_r(n) \cdot \mathcal{U}_{cpu}(n,\tau) \quad (10)$$

where $P_r(n_j) = P_{max}(n) - P_{\text{idle}}(n)$. $P_{max}(n)$ is the maximum power consumption of node n, and $\alpha(n)$ and $\beta(n)$ are the coefficients for low (i.e. $\leq thr(n)$) and high (i.e. $> thr(n)$) CPU load. Concerning $P_{net}(n,\tau)$, it is calculated for each data transfer from node n at time instant τ. We assume a linear relationship between network utilization and power consumption of data transfer, defined as $\mathcal{U}_{net}(n,m) \stackrel{\text{def}}{=} \frac{bw(n,m,\tau)}{bw_{max}(n,m)}$ where m is one of the nodes with whom n communicates at time τ. $P_{net}(n,\tau)$ is defined as,

$$P_{net}(n,\tau) \stackrel{\text{def}}{=} \sum_{m \in T(n,\tau)} \mathcal{U}_{net}(n,\tau) \cdot \gamma_{net}(n) + K_{net}(n), \quad (11)$$

where $\gamma_{net}(n)$ is a hardware-related coefficient modeling the relationship between instantaneous power and $\mathcal{U}_{net}$, $K_{net}(n)$ is a hardware-related constant and $T(n,\tau)$ is the set of nodes with whom n is communicating at instant τ. Finally, we define the energy consumption of provisioning $\mathcal{P}$ as follows, where ℓ is the infrastructure lifetime.

$$\mathcal{E}(\mathcal{P}) = \sum_{n \in \mathcal{C}} \int_0^{\ell} P_n(n,\tau)\, d\tau. \tag{12}$$

4.5 Reliability

We measure the reliability of provisioning through the joint failure probability of the provisioned ENs. For simplicity, hardware, software, and network failures affecting a node $e_k \in \mathcal{C}_e$, are channeled through a single unavailability rate, $\Pr(f_k)$, that is downtime divided by total time. We further define a joint failure probability (JFP) for all provisioned ENs. JFP, $\Pr(f_{\mathcal{P}})$, can be stated in different ways based on the availability definition of the deployed service. For instance, in the active-standby replication given in Equation 13, the service is assumed available unless all deployments fail since a standby deployment takes over when the active one fails. In load sharing replication, however, all deployments are active and share the workload. As given in Equation 14, the service is available as long as at least m deployments out of $m+n$ are available. For safety-critical services, availability might even mean that each provisioned EN is failure-free, given in Equation 15 as a special case of 14 with $n = 0$.

$$\Pr(f_{\mathcal{P}}) = \Pr\left(\bigcap_{k \in \mathcal{C}_e^{\mathcal{P}}} f_k\right) \tag{13}$$

$$\Pr(f_{\mathcal{P}}) = \Pr\left(\bigcup_{\substack{D \subseteq \mathcal{C}_e^{\mathcal{P}} \\ |D| = n+1}} \bigcap_{k \in D} f_k\right) \tag{14}$$

$$\Pr(f_{\mathcal{P}}) = \Pr\left(\bigcup_{k \in \mathcal{C}_e^{\mathcal{P}}} f_k\right) \tag{15}$$

We utilize Equation 14 in this work, which is the most general form. Finally, the reliability of a provisioning is defined as the complement of its JFP; i.e., $\mathcal{R}(\mathcal{P}) = 1 - \Pr(f_{\mathcal{P}})$.

5 ARES METHOD

The goal of ARES is to compute a trade-off between three objectives: transmission time, energy consumption, and reliability. As previously explained in Figures 3 and 4, ARES first identifies a Pareto-front containing a set of trade-off solutions between transmission time and energy consumption (i.e. candidate solutions); then, in the second stage, it augments the solutions in this set to obtain a more reliable solution, possibly at the cost of other two objectives.

5.1 Stage-1: Transmission Time and Energy

The goal of this first phase is to find a set of non-dominated solutions for the provisioning problem that minimize both latency and energy consumption. This set of solutions is called Pareto-set [20], [21]. A Pareto-set can be found with multi-objective meta-heuristics, such as MOPSO [22] and NSGA-II [20]. We employ NSGA-II meta-heuristic, due to the better performance in comparison with other meta-heuristics [23]. The input of the algorithm is (1) the set of sensors $\mathcal{S}$, (2) the network links available between each node $\mathcal{L}$, (3) the set of cloud nodes $\mathcal{C}_c$, (4) the map $\mathcal{M}(m,n)$ of the area and (5) the set of admissible coordinates for edge nodes $\mathcal{L}_e$. The pseudocode of our NSGA-II based method is described in Algorithm 1. The parameters used by our optimization are summarized in Table 2.

Algorithm 1 Transmission time and energy optimization.

```
1: function ARES–STAGE1(pSize, I, L_e, cProb, mProb, maxIter)
2:     F_0 ← generatePopulation(pSize, S, L, C_c, M(m,n), L_e)
3:     evaluateFitness(S ∪ C ∪ F_0)
4:     nIter ← 0
5:     while nIter < maxIter do
6:         F^0_nIter ← crossover(F_nIter, cProb)
7:         F^1_nIter ← mutate(F^0_nIter, mProb)
8:         evaluateFitness(F_nIter,1)
9:         F_nIter ← selection(F^0_nIter, F_nIter1)
10:        nIter ← nIter + 1
11:    end while
12:    return F_nIter
13: end function
```

Provisioning solution: According to our problem definition, e_k selected in a deployment $\mathcal{D}^e$ can be placed in a limited set of possible locations, $\mathcal{L}_e$. Let $l_0, l_1, \ldots, l_{|\mathcal{L}_e|}$ be the admissible locations in $\mathcal{L}_e$. We define then each $\mathcal{D}^e$ as a binary vector of size $|\mathcal{L}_e|$, such that

$$\mathcal{D}^e[i] = \begin{cases} 1 \iff l_i \text{ is selected to place an EN} \\ 0 \text{ otherwise.} \end{cases} \tag{16}$$

Generation of initial population: The generation of the initial population is described by Algorithm 2. We randomly generate $pSize$ provisionings ad described in Algorithm 2. Each provisioning is initialized as follows: first, we initialize $\mathcal{D}^e$ and $\mathcal{C}_e^{\mathcal{P}}$ to $\emptyset$ (line 5); Then, the algorithm iterates over the admissible coordinates for ENs(lines 6-13) and decides according to a random variable (line 7) whether to place an EN in the selected (i,j) coordinates. Initialization of EN (line 8) depends on infrastructure specifications.

Algorithm 2 Generation of initial population.

```
1: function GENERATEPOPULATION(pSize, I, M(m,n), L_e)
2:     F ← ∅
3:     p, i, k ← 0
4:     for p < pSize do
5:         C^P_e ← ∅
6:         for i < |L_e| do
7:             if rand() < 0.5 then
8:                 e_k ← initEdgeNode(I)
9:                 D_e(e_k) ← l_i
10:                C^P_e ← C_e ∪ e_k
11:                k ← k + 1
12:            end if
13:        end for
14:        P_p ← ⟨I, M_{m,n}, D_e, ⟩
15:        F ← F ∪ P_p
16:    end for
17:    return F
18: end function
```

Fitness evaluation: In this phase, we evaluate the fitness of each provisioning according to the two selected objectives: $\mathcal{T}(\mathcal{P})$ and $\mathcal{E}(\mathcal{P})$. These values are then stored for each $\mathcal{P}$ in the population $\mathcal{F}$ and used in the following crossover and selection phases.

TABLE 2
Multi-Objective optimization parameters.

Parameter	Value
Crossover type	Uniform
Crossover probability	0.7
Mutation probability	$\frac{1}{\lvert\mathcal{C}_e\rvert}$
Population size	100
Number of iteration	300

Crossover: Two solutions (parents) in the population $\mathcal{F}$ are combined to generate two new ones (offspring). We set the crossover probability to 0.7, which allows exploration of the whole search. Parents are selected using Binary Tournament Selection. After selection, a random number c is generated: if c is less than crossover probability, the two parents are combined using a uniform crossover to generate new offspring; otherwise, the two parents are returned.

Mutation: The mutation operator works by flipping a bit in the solution. First, it picks a random cell in the map: if there is no EN in that cell, it adds a node $e_{|\mathcal{C}_e|+1}$ to the solution. If there is a node instead, it removes it from the current solution. We set mutation probability to $\frac{1}{|\mathcal{C}_e|}$, to ensure that at most one placement on average is changed.

Selection: In this phase, we select the best $pSize$ solutions. Selection criteria are ranking and crowding distance, described in [20]. Selection is performed in the union of results from crossover and mutation, $\mathcal{F}^0_{nIter}$ and $\mathcal{F}^1_{nIter}$.

5.2 Stage-2: Reliability Optimization

The second stage takes the set of candidate solutions, $\mathcal{F}$, as input from the first stage. If at least one of the solutions, $S \in \mathcal{F}$, already fulfills the reliability requirement, it can be used without further search. Otherwise, nearby ENs (i.e. neighborhood sets) are taken into account. Since the failures at the different ENs can be correlated regardless of the geographical placement, the combination of the individually most reliable nodes from each neighborhood sets might not always be the most reliable deployment strategy. Consequently, we need to compute the joint failure probability of each possible combination. A trivial example with only two ENs in a candidate solution is demonstrated in Figure 6. Here, the initial solution is found unreliable and five and six additional ENs in the two cells are evaluated in combination. Finally, both ENs are replaced with alternatives in their neighborhood sets to improve reliability. In this example, further search in neighbor cells is not necessary.

The pseudo-code description of stage-2 is presented in Algorithm 3. For a candidate solution, we populate the neighborhood sets, N^S_k, for each chosen EN, $e_k \in S$. Initially, the sets contain only the chosen ENs (line 4). We gradually expand the set with the nodes in the same cell, immediate neighbor cells, neighbor cells of distance two (i.e. neighbors of neighbors), and so on (line 24). At each step, alternative solutions are generated by selecting exactly one node from each neighborhood set (line 13). The search stops when a solution fulfills the reliability requirement, ρ (line 16). Therefore, the algorithm can identify a reliable deployment without excessive modification to the solutions that are estimated to be Pareto optimal in terms of transmission time and energy. We omit the selection process for the next alternative node from each neighborhood set for brevity. Depending on the function $nextAlternative()$, they can be evaluated in descending order of reliability to shorten the search, or ascending order of distance from the chosen node to minimize the extent of modification.

Algorithm 3 Reliable deployment search.

```
1: function ARES–STAGE2(F, ρ)
2:     for all S ∈ F do
3:         for all e ∈ S do
4:             N_e^S ← e
5:         end for
6:     end for
7:     d ← 0
8:     while d < d_max do
9:         for all S ∈ F do
10:            while nextAlternativeExist() do
11:                C ← ∅
12:                for all e_k ∈ S do
13:                    C ← C ∪ nextAlternative(N_k^S)
14:                end for
15:                if jointFailureProbability(C) < ρ then
16:                    return C
17:                end if
18:            end while
19:        end for
20:        for all S ∈ F do
21:            for all e_k ∈ S do
22:                (x, y) ← D_e(e_k)
23:                for all (i, j) ∈ neigh(x, y, d) do
24:                    N_k^S ← N_k^S ∪ M_{m,n}(i, j)
25:                end for
26:                nextAlternativeReset(N_k^S)
27:            end for
28:        end for
29:        d ← d + 1
30:    end while
31: end function
```

JFP estimation: The reliability evaluation of candidate solutions relies on the JFP value (line 15). However, computing exact JFP requires the computation and storage of an exponential number of probability values to the number of ENs [24]. To that end, we employ dynamic Bayesian networks (DBN) which have been shown effective in estimating spatial and temporal failure dependencies in edge computing systems [8]. A DBN identifies the strongest dependencies between random events so that only those that are significant to the JFP are taken into account. In Figure 7a, the structure of an example DBN with three ENs in two time steps is visualized. Here, the arrows indicate the direction of dependency. For instance, the failure of node 3 at time t (f^t_3)

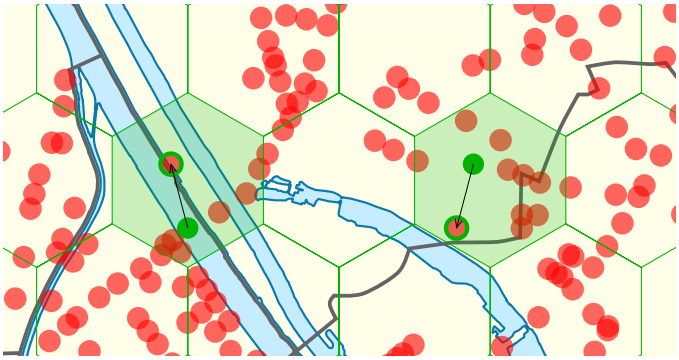
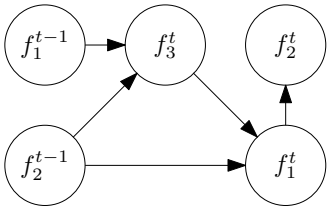


Fig. 6. Geographical distribution of a solution from stage-1 (solid green circles), evaluated solutions in stage-2 (red circles that lie in hexagonal areas), and the final reliable solution (hollow green circles).



(a) Structure

f_2^{t-1}	f_3^t	$\Pr(f_1^t)$	$\Pr(\neg f_1^t)$
T	T	0.10	0.90
T	F	0.08	0.92
F	T	0.06	0.94
F	F	0.05	0.95

(b) Parameters

Fig. 7. An example dynamic Bayesian network for three edge nodes.

or node 2 at time $t-1$ (f_2^{t-1}) causes a consequent failure of node 1 at time t (f_1^t). The strength of these dependencies are recorded in conditional probability tables (CPT). An example CPT for f_1^t is given in Figure 7b, which shows that the occurrence of both cause events yields the highest failure probability at e_1. We train the DBN structure and parameters automatically from past failure traces.

The independence assumption of Bayesian networks states that a variable is conditionally independent of its non-descendants, given its parents. This allows us to factorize the joint distribution of a set of variables by conditioning each variable only on its parents in the DBN. Accordingly, the inner intersection in Equation 14 can be estimated as,

$$\Pr\left(\bigcap_{k\in D} f_k\right) = \prod_{k\in D} \Pr\left(f_k \,\middle|\, \bigcap_{p\in par(f_k)} f_\gamma\right)$$

Here, $par()$ is a function that returns the parents of an event in the DBN (i.e. its causes). Since all these conditional properties are already available in the corresponding CPTs, estimation of the JFP reduces to a series of arithmetic operations assuming the DBN is already trained offline.

5.3 Complexity Analysis

We describe now the complexity of two optimization stages. Stage-1 is based on NSGA-II metaheuristics, whose complexity is determined by three parameters: number of iterations $maxIter$, the number of objectives, and the complexity of a single iteration. The complexity of a single iteration depends on the complexity of each phase. First, the generation of the initial population is performed by randomly setting to 1 different bits of each solution. Solutions have size $|\mathcal{L}_e|$, and p solutions are generated, giving a complexity of $\mathcal{O}(p \cdot |\mathcal{L}_e|)$. Crossover operator generates p new solutions, selecting two solutions and combining them using a uniform crossover, whose complexity is $\mathcal{O}(|\mathcal{L}_e|)$. Since the selection of parents is performed p times, the complexity of crossover phase is $\mathcal{O}(p \cdot p \cdot |\mathcal{L}_e|) = \mathcal{O}(|\mathcal{L}_e| \cdot p^2)$. Concerning the mutation phase, the bit flip requires $\mathcal{O}(1)$, and it is performed at most p times, therefore complexity is $\mathcal{O}(p)$. Finally, selection can be considered as a sorting of set $\mathcal{F}^0_{nIter} \cup \mathcal{F}^1_{nIter}$, whose size is $2p$, therefore its complexity accounts for $\mathcal{O}(p \cdot \log p)$, assuming that we use quicksort as sorting algorithm. Since the highest term is $\mathcal{O}(p^2)$, we consider each iteration to have a $\mathcal{O}(p^2)$ complexity. Since we consider two objectives, this brings us to the final complexity of $\mathcal{O}(2 \cdot maxIter \cdot populationSize^2)$.

Stage-2 iterates over solutions that include exactly one node from each neighborhood. The sizes of the neighborhoods increase by $6d$ hexagons at each iteration d so the size at iteration d is $3d(d+1)$. Therefore, the number of augmented solutions originating from a candidate solution S can be computed as in Equation 17. Accordingly, the worst-case time complexity of Algorithm 3 is $\mathcal{O}(d^{2|S|})$.

$$\prod_{k=1}^{|S|} \binom{|N_k|}{1} = \prod_{k=1}^{|S|} |N_k| = \prod_{k=1}^{|S|} 3d(d+1) \tag{17}$$

TABLE 3
Hardware configuration.

$n \in \mathcal{C}$	CPU	MIPS
c-*	64	15
e-*	4	2

TABLE 4
Energy coefficients for Equations 7-11, from [19].

Coefficient	Value
α	5.29
β	0.68
γ_{3g}	$0.025e-6$
$\mathcal{K}_{3g}$	$3.5e-6$
γ_{wifi}	$0.007e-6$
K_{wifi}	$5.9e-6$
P_{idle}	501
P_{max}	840
$thr(n)$	0.12

6 Experimental Setup

We make use of real traffic light locations in Vienna [13] to simulate the above-described WSN. The dataset contains records for 1,369 traffic lights illustrated in Figure 1. For each location, we deploy two camera sensors, as in `InTraSafEd5G` motivational use cases, and at most one EN, according to results of the ARES method. Our evaluation is performed through simulations. After investigating different edge simulators such as iFogSim [25] and EdgeCloudSim [26], we based our simulation on SLEIPNIR[3], described in [18]. SLEIPNIR simulator runs on Apache Spark, which allows it to easily scale according to underlying computational resources. Moreover, it provides validated models for edge/cloud infrastructure. We extend this version by adding support for (1) IoT devices, (2) edge provisioning, and (3) reliability models. For the multi-objective optimization algorithms, we employed JMetal v5 library [27], the de-facto standard for multi-objective metaheuristics.

6.1 Computational Nodes

We assume that the CPU and MIPS of computational edge nodes do not change at each run of the simulation. This is because, in real-world scenarios, the hardware configuration of computational nodes changes rarely during one single application execution. We assume that ENs have less capabilities than cloud nodes [28]. The hardware specifications and hardware resources cost for each node are shown in Table 3. Energy consumption of computational nodes for the Equations 7-11, are given in Table 4. For the values of $\mathcal{U}_{cpu}(n,\tau)$, we use a uniform distribution. We simulate computational load using traces coming from our `InTraSafEd5G` use case. Traces include execution time for object detection using MobileNet-SSDv2 on each frame.

6.2 Network Infrastructure

Due to the unreliability of connections in WSN, we need to accurately model the unreliable connections between

3. https://github.com/vindem/sleipnir

TABLE 5
Network availability distribution.

Connection	Availability	QoS profile		Probability
		Latency (ms)	Bandwidth (Mbps)	
3G	0.75	54	7.2	0.9957
		∞	0	0.043
WiFi	0.25	15	32	0.9
		15	4	0.09
		∞	0	0.01

sensors and computational nodes. We model $latency(s, n)$ and $bw(s, n)$ as random variables. The distribution of $latency(s, n)$ and $bw(s, n)$ depends on the connection available between s and n. We assume that two different connections are available: 3G and WiFi. The availability of connection is determined by $wifi_{av}$ uniform random variable. If both are available during the execution, the one with the lowest latency is selected. Probability distributions are defined in [29] and summarized in Table 5. $latency(s, n) = \infty$ and $bw(s, n) = 0$ means that no connection is available. Coefficients are summarized in Table 4, where γ_{3g},K_{3g} and γ_{wifi},K_{wifi} refer, respectively, to γ_{net} and K_{net} when using a 3G or WiFi. Data transfer size is modeled by an exponential random variable whose λ parameter is set to 1mb, which is the average size of frames captured by cameras.

6.3 Node Failures

To the best of our knowledge, there does not exist an edge computing reliability data set that is available to the research community at present. This is because of not only the novelty of the technology, but also the obstacles to making workload traces of commercial systems publicly available, such as competitive concerns, privacy obligations, and hardness of data anonymization [30]. However, failure characteristics of the edge infrastructure is not entirely different from that of other widely distributed and virtualized computing systems [8], for which failure data sets are available. Among various deployment strategies for edge computing, we focus on the most widely distributed case of employing end devices. Therefore, we utilize failure traces of UC Berkeley SETI@home volunteer computing project [31] collected from 226,208 personal computers between April 1, 2007 and January 1, 2009. This data set models an edge infrastructure that is formed by devices with high churn and low reliability as expected from a cost-efficient, large-scale urban deployment. We assume that the service is available unless more than 20% of the deployed nodes are in failure (i.e. $m = 4n$) as defined in Equation 14.

6.4 Baseline Algorithms

ARES Stage-1 Only (STG1) estimates the Pareto front as described in Section 5.1 upon transmission time and energy consumption. Among the candidate solutions on the front, the one with the highest expected reliability is chosen by executing only the iteration-0 of the stage-2. The candidate solutions are not augmented so as to improve reliability.

Joint Failure Probability (JFP), on the other hand, focuses solely on reliability. We compare ARES to the JFP algorithm from previous work [8], which computes the smallest subset of edge nodes that are expected to satisfy a given reliability requirement (i.e. 99.8%). This baseline ignores transmission time and energy consumption.

Three-Objective Optimization (3OBJ) enhances stage-1 of ARES with a third objective that is the expected marginal reliability of edge nodes. Thus, this baseline assumes independent failures similar to how reliability is handled in previous work [32] along with two other objectives. For our comparison, we select the best solution for each objective on the Pareto-Front computed by the algorithm and compare it with the ARES solution.

Facility Location Problem (FLP) is a single-objective algorithm aiming at minimizing energy consumption without considering latency and reliability. It finds the optimal set of nodes that minimize total energy consumption. FLP is implemented as an integer linear programming problem, following the description in [33], using the ECOS BB solver [34] implemented using Python 3.5 CVXPYmodule.

7 Numerical Results

7.1 Trade-Off Evaluation

In the first part of the evaluation, we provide an empirical analysis of the trade-off between the two stages of ARES. Figure 8 demonstrates the mean distance of the solutions from the stage-1 Pareto front with 99.9% confidence interval (CI) as the required reliability in the service level agreement (SLA) increases logarithmically. Here, distance corresponds to the variable d in Algorithm 3. 90% SLA can always be achieved without augmenting the solutions from stage-1 ($\bar{d} = 0$). This is also true with 99% SLA for the most solutions ($\bar{d} = 0.53 \pm 0.33$). As expected, stage-2 explores farther nodes to satisfy higher reliability requirements. Note that, 8 of the 100 trials did not reach 5 nines (99.999%) availability and thus are not taken into account for the last data point, which explains the flattening of the curve.

Increasing the distance causes higher energy consumption and longer transmission time as Figures 9 and 10 show. CI is also significantly larger because the augmentation of Pareto-optimal solutions results in an arbitrary impact on these two metrics. However, the impact is limited even in the worst case. Mean transmission time with 5 nines reliability is 9.27% longer than the shortest possible on the Pareto-front and mean energy consumption is only 1.47% higher than the lowest. Depending on the criticality of the service, the provider might opt for lower reliability values to further reduce the impact. For instance, 4 nines can be achieved with less than 5% time and 1% energy deterioration.

7.2 Convergence Analysis

Convergence of Stage-1: We check how Pareto-front calculated in stage-1 converges to the optimal Pareto front by gradually increase iteration number by 50, starting from 100. We use Hypervolume [35] as a quality indicator, as it is considered a measure of convergence and diversity of the whole Pareto front. The results of this evaluation are summarized in Figure 11. They show that after 300 iterations, increasing the number of iterations does not significantly improve Hypervolume, regardless of the additional time spent by the algorithm (see Figure 12), therefore we select 300 as iteration number as given in Table 2.

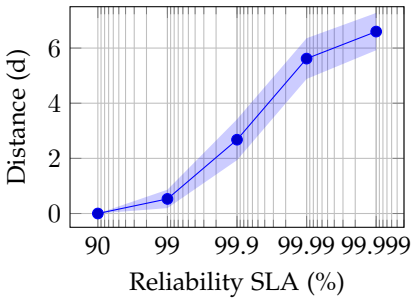


Fig. 8. Distance From Pareto vs. Reliability.

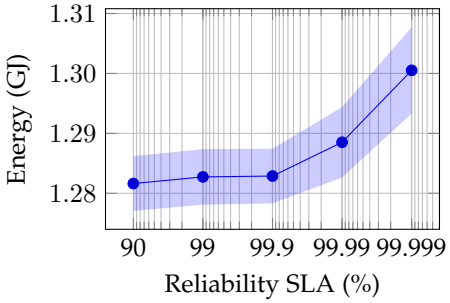


Fig. 9. Energy Consumption vs. Reliability.

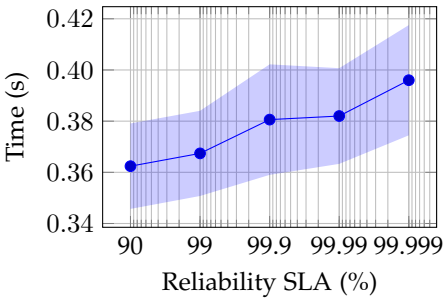


Fig. 10. Transmission Time vs. Reliability.

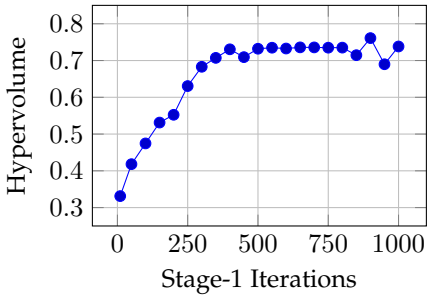


Fig. 11. Hypervolume vs. Number of Iterations.

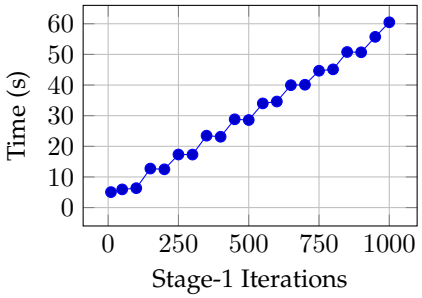


Fig. 12. Time vs. Number of Iterations.

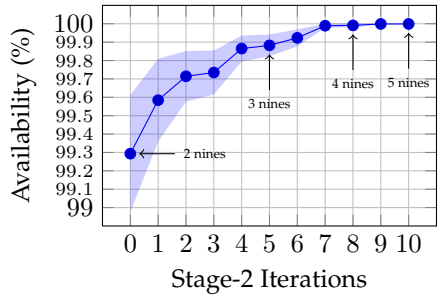


Fig. 13. Reliability vs. Number of Iterations.

Convergence of Stage-2: We analyze the convergence to 100% reliability in terms of stage-2 iterations (i.e. line 8 in Algorithm 3). Figure 13 shows the mean service availability achieved in 100 executions of stage-2 from different inputs from stage-1. Here, the shaded areas indicate 99.9% CI of the results. As the number of maximum allowed iterations (d_{max}) increases, both mean service time and confidence interval improve. The trend is strictly increasing, showing that on average every additional iteration yields better reliability than the previous one. Original solutions on the Pareto front (iteration-0) suffer 0.7% unavailability on average, yet it is possible to achieve 3 nines of availability (0.01% unavailability) only after 5 iterations or cloud-grade 5 nines of availability after 10 iterations. Each iteration of stage-2 takes sub-second time on an Intel Xeon E5-2650 processor.

7.3 Comparative Evaluation

We compare the performance of ARES to the baseline algorithms described in Section 6.4 with respect to the energy-efficiency, transmission time, and fault-tolerance. Additionally, three different iteration lengths of the ARES stage-2 are evaluated to demonstrate its flexibility. We first direct our attention to the comparison of energy efficiency based on Figures 14, 15, and 16, which respectively compare total energy consumption (Equation 12), energy consumption per data transmission (Equation 11), and the number of the ENs provisioned by each algorithm. All results are accompanied by the 99.9% CI. Here, FLP acts as a baseline that represents minimum possible energy consumption to assess how close the algorithms are to the optimum energy efficiency.

ARES is allowed to explore ENs farther from the stage-1 solution at each iteration of the stage-2. Therefore, Pareto optimal locations gradually drift resulting in a slightly higher total and per transmission energy consumption. The increase from three- to seven- iteration versions are 1.4% and 6.6% for the two metrics, respectively. We also observe that total energy consumption with ARES is only between 4.5% to 6.0% higher than the optimum (FLP). Since augmentation at each iteration can only replace the ENs with their neighbors and cannot add or remove nodes, the number of provisioned nodes does not change in ARES.

3OBJ and STG1 achieve comparable total consumption to the five- and three-iteration versions of ARES, respectively. The differences are within the 99.9% CI in Figure 14. The other two results of STG1 are also similar to three-iteration ARES. This shows that reliability optimization at stage-2 does not cause a significant deterioration in energy efficiency, especially with few iterations. 3OBJ, on the other hand, provisions 2.8 fewer ENs on average than ARES. This is not reflected in total energy consumption because consumption per transmission is 2.7% higher. The inclusion of the third objective (reliability) results in a slight (0.5%) increase in energy consumption with respect to STG1.

JFP outperforms all algorithms except energy-optimum FLP in mean cumulative energy consumption (Figure 14) because it provisions fewer (but more reliable) ENs. However, CI is comparatively very large, which indicates randomness in its energy efficiency performance. The reason is, JFP ignores the location of provisioned nodes, which also affects the energy consumption due to communication distance. This is clear in the per transmission results where it has the worst performance and again large CI.

We consider fault-tolerance and transmission time in Figures 17 and 18, respectively. In line with the energy-efficiency results, ARES suffers from a slight increase in transmission time (7.8% from 3 to 7 iterations) but achieves significant improvement in fault-tolerance (almost 25 times less unavailability) as more iterations are run. JFP success-

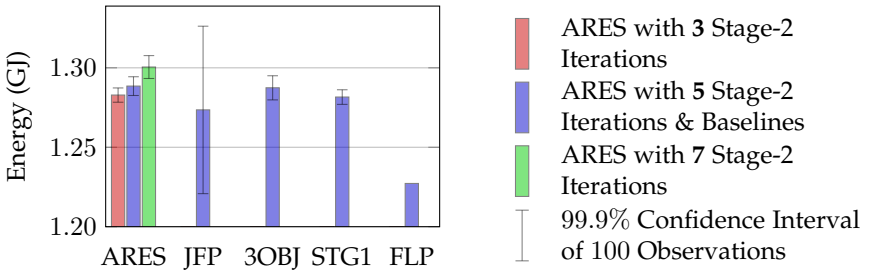


Fig. 14. Total Daily Energy Consumption.

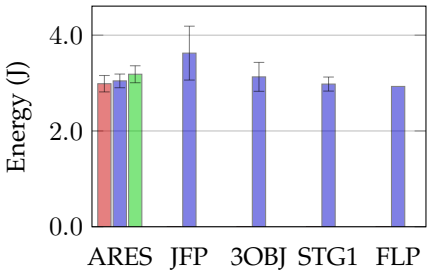


Fig. 15. Per Transmission Energy Consumption.

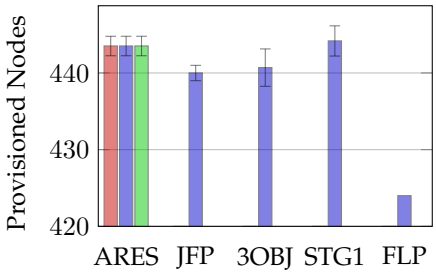


Fig. 16. Number of Provisioned Nodes.

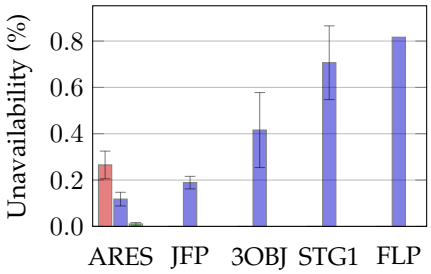


Fig. 17. Service Reliability Level.

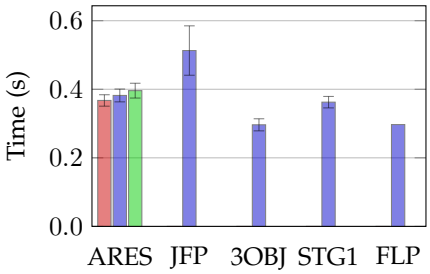


Fig. 18. Transmission Time.

fully finds provisionings with the targeted level of availability (i.e. 99.8%) with low variance. However, this results in substantially higher latency (34.3% higher than ARES on average). Five and seven iteration versions of ARES achieve both higher reliability and lower latency than JFP.

The transmission time of STG1, as its energy consumption, is comparable to three-iteration ARES, however, its unavailability is 2.7 times higher. Moreover, five- and seven-iteration versions of ARES outperform STG1 by 6 and 65 times in terms of unavailability at the cost of a marginal increase in responsiveness and energy-efficiency. Therefore, we conclude that simply choosing the most reliable solution on the Pareto front is not sufficient and solution augmentation (stage-2) is imperative. FLP has higher unavailability than STG1 because it provisions significantly fewer ENs.

In contrast to energy-efficiency results, 3OBJ outperforms STG1 in transmission time with 18.2% less latency. This is because 3OBJ solutions contain more ENs than the STG1 ones, in order to satisfy the reliability trade-off that is not considered in STG1. In terms of fault-tolerance, on the other hand, the inclusion of the third objective results in 40% less unavailability than STG1. However, even this improved reliability value is at least two times higher than those of JFP and ARES because 3OBJ only evaluates marginal failure probabilities and ignores the failure dependencies between edge nodes. The results clearly show that correlated failures play an important role in the reliability of edge provisioning.

8 Related Work

A two-stage optimization method coding is defined in [36], considering as objectives latency, reliability, and storage size. The method, however, targets erasure coding, and it is focused on Cloud resources rather than ENs. More similar to our work, [37] focuses on optimization of ENs resource allocation, considering as parameters QoS and reliability, without considering energy. In [18], the authors focused on multi-objective offline provisioning, focusing on energy and cost efficiency rather than on reliability. Resource management on ENs has been extensively discussed by [29], [32], [38], [39], focusing more on the application and user perspective than on the provider side. From the WSN sides, several optimization methods have been proposed, either on the side of routing [40] or for the base stations provisioning in mobile cellular networks [16], without considering energy efficiency and reliability of data processing.

Although essential for its success, failure resilience in edge computing is still an open issue [41]. An early discussion of reliability challenges in fog computing is presented in [42]. However, few attempts are made to address these challenges. Aral and Brandic introduce a technique that exploits causal relationships between different types of failure and channel all QoS related parameters through VM availability [43], [44]. Nebula [38], an edge-based computation and storage architecture, handles fault-tolerance of compute nodes via re-execution. Although data is replicated, availability is not a factor in replica site selection. Cloud visitation platform [45], which copes with the hardware heterogeneity problem in federated clouds and fog via hardware awareness, solves failure resilience only at the local level. When a server fails, deployed applications are migrated to another one, possibly in a different node. Cardellini et al. [39] extend well known distributed stream processor, Apache Storm, by adding QoS awareness capability. The proposed scheduler chooses resources based on latency as well as utilization and availability. Here, the recent availability of nodes is used instead of predicting future availability. FogStore [46], a distributed data store, handles replica and consistency management. As only data blocks are replicated, the focus of this work is on read and write latency. Recently, a recovery scheme for edge computing failures is proposed in [47]. However, only the failures that are caused by overloaded

resources are considered. Traffic data is monitored to detect such nodes and their load is distributed. Odin platform [48] is a practical application of fault-tolerance for distributed servers. It detects failures and creates backups in CDNs.

The problem of energy-efficiency in WSN has been discussed by [49], mostly from the network side and not from the data analytics side. The advantage of combining edge analytics and WSN is discussed in [50], [51], [52], without considering fault-tolerance and energy-efficiency. In [53], authors discuss a clustering algorithm using ENs to foster uniform energy utilization over the WSN, while in [40] the problem is discussed from the point of view of routing inside the network. Provisioning of ENs considering analytics workload and energy-efficiency among different objectives is discussed in [18], but in the context of mobile offloading. Conversely, [54] focuses on the energy-efficiency and fairness of processing applications in mobile WSN. In the context of industrial WSN, [55] proposes the use of fog computing to achieve energy-efficiency.

9 Conclusion and Future Work

In this work, we propose ARES, a two-stage optimization method for offline provisioning of ENs. In the first stage, ARES uses NSGA-II multi-objective metaheuristic to obtain a set of trade-off solutions for transmission time and energy consumption; then, in the second stage, it improves the solutions obtained in the first stage to achieve the desired level of reliability. By means of this two-stage optimization, ARES is capable of achieving sustainable and reliable provisioning of ENs in an urban area combining the reliability benefits of considering correlated failures with energy and transmission time optimization. We evaluate the results of our method in comparison to four state-of-the-art provisioning algorithms using data coming from a real-world setting. The results show that ARES is capable of achieving a better trade-off between transmission time, energy consumption, and reliability in a significantly shorter time.

As future work, we plan to include more coordination and concurrency between the two stages, in order to further reduce execution time. Additionally, we plan to minimize the number of additional edge node provisionings and suspensions during re-optimization after some changes in environmental conditions. Finally, we plan to evaluate ARES in an online setting and improve it accordingly.

Acknowledgments

This work has been partially funded through the Rucon project (Runtime Control in Multi Clouds), Austrian Science Fund (FWF): Y904-N31 START-Programm 2015, and 5G Use Case Challenge InTraSafEd 5G (Increasing Traffic Safety with Edge and 5G) funded by the City of Vienna.

References

[1] Mordor Intelligence Editors, "Smart sensors market - growth, trends, and forecast," Mordor Intelligence LLP, Tech. Rep., 2019.

[2] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[3] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *ACM workshop on Security of adhoc and sensor networks*, 2006, pp. 169–176.

[4] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," in *ACM SIGCOMM conference on Internet measurement*, 2010, pp. 1–14.

[5] J. Gama and M. M. Gaber, *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.

[6] A. Sorniotti, L. Gomez, K. Wrona, and L. Odorico, "Secure and trusted in-network data processing in wireless sensor networks," *J. of Info. Assurance and Security*, vol. 2, no. 3, pp. 189–199, 2007.

[7] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[8] A. Aral and I. Brandic, "Dependency mining for service resilience at the edge," in *IEEE/ACM Symposium on Edge Computing (SEC)*, 2018, pp. 228–242.

[9] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[10] H. Yan, H. Huo, Y. Xu, and M. Gidlund, "Wireless sensor network based e-health system," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, pp. 2288–2295, 2010.

[11] D. Chen, Z. Liu, L. Wang, M. Dou, J. Chen, and H. Li, "Natural disaster monitoring with wireless sensor networks: A case study of data-intensive applications upon low-cost scalable systems," *Mobile Networks and Applications*, vol. 18, pp. 651–663, 2013.

[12] G. S. Menon, M. V. Ramesh, and P. Divya, "A low cost WSN for water quality monitoring in natural water bodies," in *IEEE Global Humanitarian Technology Conference (GHTC)*, 2017, pp. 1–8.

[13] Vienna Municipal Department 33, "Traffic lights with/without audible signal devices in Vienna," https://www.data.gv.at/, 2019, Open Data Österreich.

[14] S. B. Letaifa, "How to strategize smart cities: Revealing the smart model," *J. of business research*, vol. 68, no. 7, pp. 1414–1419, 2015.

[15] S. Fu and C.-Z. Xu, "Exploring event correlation for failure prediction in coalitions of clusters," in *ACM/IEEE conference on Supercomputing*, 2007, p. 41.

[16] S. Govindasamy and I. Bergel, "Uplink performance of multi-antenna cellular networks with co-operative base stations and user-centric clustering," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2703–2717, 2018.

[17] C. Pahl, S. Helmer, L. Miori, J. Sanin, and B. Lee, "A container-based edge cloud paas architecture based on raspberry pi clusters," in *IEEE International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, 2016, pp. 117–124.

[18] V. De Maio and I. Brandic, "Multi-objective mobile edge provisioning in small cell clouds," in *ACM/SPEC International Conference on Performance Engineering*. ACM, 2019, pp. 127–138.

[19] V. De Maio, G. Kecskemeti, and R. Prodan, "An improved model for live migration in data centre simulators," in *International Conference on Utility and Cloud Computing*. ACM, 2016, pp. 108–117.

[20] 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.

[21] H. M. Fard, R. Prodan, and T. Fahringer, "Multi-objective list scheduling of workflow applications in distributed computing infrastructures," *Journal of Parallel and Distributed Computing*, vol. 74, no. 3, pp. 2152 – 2165, 2014.

[22] J. Hao, Z. Jin-hua, and C. liang jun, "Multi-objective particle swarm optimization algorithm based on enhanced ϵ-dominance," in *IEEE Int'l Conf. on Eng. of Intelligent Systems*, 2006, pp. 1–5.

[23] C. Zambrano-Vega, A. J. Nebro, J. García-Nieto, and J. F. A. Montes, "Comparing multi-objective metaheuristics for solving a three-objective formulation of multiple sequence alignment," *Progress in Artificial Intelligence*, vol. 6, pp. 195–210, 2017.

[24] N. Friedman, K. Murphy, and S. Russell, "Learning the structure of dynamic probabilistic networks," in *Conference on Uncertainty in artificial intelligence*. Morgan Kaufmann, 1998, pp. 139–147.

[25] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments," *CoRR*, vol. abs/1606.02007, 2016.

[26] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," in *Int'l Conf. on Fog and Mobile Edge Computing*, 2017, pp. 39–44.

[27] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.

[28] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Workshop on Mobile cloud computing*, 2012, pp. 13–16.
[29] A. Brogi, S. Forti, and A. Ibrahim, "How to best deploy your fog applications, probably," in *IEEE International Conference on Fog and Edge Computing (ICFEC)*, 2017, pp. 105–114.
[30] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Obfuscatory obscanturism: making workload traces of commercially-sensitive systems safe to release," in *IEEE Network Operations and Management Symposium*, 2012, pp. 1279–1286.
[31] B. Javadi, D. Kondo, J. Vincent, and D. Anderson, "Mining for statistical availability models in large-scale distributed systems: An empirical study of SETI@home," in *ACM Int'l Symp. on Modelling, Analysis and Simulation of Computer and Telecomm. Systems*, 2009.
[32] V. De Maio and D. Kimovski, "Multi-objective scheduling of extreme data scientific workflows in fog," *Future Generation Computer Systems*, vol. 106, pp. 171 – 184, 2020.
[33] A. Aral and T. Ovatman, "A decentralized replica placement algorithm for edge computing," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 516–529, 2018.
[34] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An socp solver for embedded systems," in *European Control Conference*. IEEE, 2013, pp. 3071–3076.
[35] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, 2006.
[36] N. Saurabh, D. Kimovski, F. Gaetano, and R. Prodan, "A two-stage multi-objective optimization of erasure coding in overlay networks," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2017, pp. 150–159.
[37] H. Sun, H. Yu, G. Fan, and L. Chen, "Qos-aware task placement with fault-tolerance in the edge-cloud," *IEEE Access*, pp. 1–1, 2020.
[38] M. Ryden, K. Oh, A. Chandra, and J. Weissman, "Nebula: Distributed edge cloud for data intensive computing," in *IEEE International Conference on Cloud Engineering*, 2014, pp. 57–66.
[39] V. Cardellini, V. Grassi, F. L. Presti, and M. Nardelli, "On QoS-aware scheduling of data stream applications over fog computing infrastructures," in *IEEE Symposium on Computers and Communication*, 2015, pp. 271–276.
[40] C. Walker and A. Al-Anbuky, "LED-WSN: Light weight edge computed dynamic wireless sensor network routing protocol," in *International Telecommunication Networks and Applications Conference*, 2017, pp. 1–8.
[41] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.
[42] H. Madsen, B. Burtschy, G. Albeanu, and F. Popentiu-Vladicescu, "Reliability in the utility computing era: Towards reliable fog computing," in *International Conference on Systems, Signals and Image Processing*. IEEE, 2013, pp. 43–46.
[43] A. Aral and I. Brandic, "Quality of Service Channelling for Latency Sensitive Edge Applications," in *IEEE International Conference on Edge Computing*, 2017, pp. 166–173.
[44] ——, "Learning spatiotemporal failure dependencies for resilient edge computing services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, 2021.
[45] M. Zhanikeev, "A cloud visitation platform to facilitate cloud federation and fog comp." *Computer*, vol. 48, no. 5, pp. 80–83, 2015.
[46] R. Mayer, H. Gupta, E. Saurez, and U. Ramachandran, "Fogstore: Toward a distributed data store for fog computing," in *IEEE Fog World Congress*, 2017, pp. 1–6.
[47] D. Satria, D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Future Generation Computer Systems*, vol. 70, pp. 138–147, 2017.
[48] M. Calder, R. Gao, M. Schröder, R. Stewart, J. Padhye, R. Mahajan, G. Ananthanarayanan, and E. Katz-Bassett, "Odin: Microsoft's scalable fault-tolerant CDN measurement system," in *USENIX Symp. on Networked Systems Design and Impl.*, 2018, pp. 501–517.
[49] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104 – 122, 2014.
[50] K. Bhargava, S. Ivanov, W. Donnelly, and C. Kulatunga, "Using edge analytics to improve data collection in precision dairy farming," in *IEEE Conference on Local Computer Networks Workshops*, 2016, pp. 137–144.
[51] K. Bhargava, S. Ivanov, D. McSweeney, and W. Donnelly, "Leveraging fog analytics for context-aware sensing in cooperative wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 15, no. 2, pp. 1–35, 2019.
[52] S. Yang, "Iot stream processing and analytics in the fog," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 21–27, 2017.
[53] M. Venkateswarlu, A. Kandasamy, and K. Chandrasekaran, "An energy-efficient clustering algorithm for edge-based wireless sensor networks," *Procedia Computer Science*, vol. 89, pp. 7–16, 2016.
[54] Z. Sheng, C. Mahapatra, V. M. Leung, M. Chen, and P. Sahu, "Energy efficient cooperative computing in mobile WSNs," *IEEE Transactions on Cloud Computing*, vol. 6, no. 01, pp. 114–126, 2018.
[55] K. Suto, H. Nishiyama, N. Kato, and C. Huang, "An energy-efficient and delay-aware wireless computing system for industrial WSNs," *IEEE Access*, vol. 3, pp. 1026–1035, 2015.

Atakan Aral is a Postdoctoral Research Fellow at Vienna University of Technology. He received a dual MSc degree in Computer Science and Engineering from Politecnico di Milano in 2011 and Istanbul Technical University (ITU) in 2012, and a PhD degree in Computer Engineering from ITU in 2016. His research interests center around resource management for geo-distributed and virtualized computing systems such as intercloud and edge computing, as well as optimization of the edge computing architecture for AI services.

Vincenzo De Maio received his Ph.D. in 2016 at the University of Innsbruck, Austria. His research in the area of parallel and distributed systems comprises energy-aware Cloud computing and scheduling. Since 2017, he is a postdoctoral researcher at the Institute of Information Systems Engineering of the Vienna University of Technology. He authored different conference and journal publications on the topic of energy efficiency and modeling for Cloud and Edge computing.

Ivona Brandic is a Professor at the Vienna University of Technology. In 2015, she was awarded the FWF START prize, the highest Austrian award for early career researchers. She received her PhD degree in 2007 from Vienna University of Technology. In 2011, she received the Distinguished Young Scientist Award from the Vienna University of Technology for her project on the Holistic Energy Efficient Hybrid Clouds. Her main research interests are cloud computing, large scale distributed systems, energy efficiency, quality of service, and autonomic computing.