Validating the Optimization of a Building Occupancy Monitoring Software System

Jalil Boudjadar and Simon Thrane Hansen
Department of Electrical and Computer Engineering, Aarhus University, Denmark

Keywords: Buildings Occupancy Analysis, State Estimation, Monitoring Software, Formal Verification.

Abstract: Real-time occupancy information is key to track and optimize the resource usage in buildings. With the ultimate goal to reduce energy consumption and improve indoor environment quality, different occupancy monitoring solutions have been introduced in the literature. However, these solutions can be part of the original problem due to its high energy consumption. This paper proposes the design and validation of a knowledge-driven energy-efficient system to monitor and analyze buildings occupancy. Such a system is in fact an optimization of the actual software solution (Office Master 3000) used to monitor buildings occupancy in an academic institution. The key contributions are: 1) *Optimization of the actual solution*: using the knowledge about the activity expected to take place, our system proactively identifies the minimal sensors data relevant to the actual state to confirm/deny such an activity by which non-substantial sensors are operated with loosened frequency; 2) *Formal validation*: using UPPAAL model checker we prove that the original functional and non-functional properties are maintained post-optimization. The proposed system is implemented in C++, tested and validated. The results demonstrate that our optimization reduces sensors energy consumption by up to 31% while maintaining high accuracy (84%) in identifying the occupancy states and activities.

1 INTRODUCTION

With the ever increasing buildings energy demand and cost, it is paramount to monitor and optimize the energy consumption of the different loads and computing components embedded in buildings (Lasla et al., 2019; Boudjadar et al., 2014). To conduct such an optimization, a highly reliable tracking of the actual occupancy state of building spaces is needed. The building state refers to the activities, events and potentially the number of people occupying a building space at a given time point. Building occupancy monitoring systems (BOMS) enable to track the state of building spaces in real-time (Azimi and O'Brien, 2022; Elkhoukhi et al., 2018; Shokrollahi et al., 2024; Baerentzen et al., 2023; Tien et al., 2022). The knowledge generated by BOMS provides a solid and informative ground for buildings management to identify when and where resource utilization can be reduced, so that to achieve a better space allocation and efficient resources usage (Salimi and Hammad, 2019; Jabirullah et al., 2021).

Commonly, state of the art BOMS rely on *Big Data* analysis where *massive* quantities of sensor data (video, audio, step panels, etc.) need to be collected and run through classification, prediction and optimization engines (Zhang et al., 2019) (Jiang and Yin, 2015). It is expensive to maintain the real-time aspect of such a brute-force approach to occupancy analysis (Elkhoukhi et al., 2018), due to massive data processing, and can result in relatively low accuracy, particularly if the input data blinds out the data related to the cases with a strong potential to optimization (Salimi and Hammad, 2020; Pan et al., 2014). Moreover, it can end up with high energy consumption due to the cost of collecting and processing massive data (Costenaro and Duer, 2012).

Different attempts to optimize the energy consumption of BOMS and improve its decision making have been proposed in the literature (Lou et al., 2017; Rai et al., 2015; McKenna et al., 2015; Dai et al., 2020; Ahmad et al., 2021; Zhang et al., 2022; Jiang et al., 2022). However, different challenges have emerged: 1) it is not trivial to design high fidelity models to enable systematic state computation (Austin et al., 2020); 2) an automated semantics-driven occupancy analysis delivering high confidence decisions in real-time is expensive computation-wise; 3) optimizing the performance and energy efficiency of a BOMS can lead to violate safety properties and degrade reliability metrics given that such metrics can be conflicting with the performance (Teng et al., 2021).

This paper proposes the optimization of a state of

Boudjadar, J., Hansen and S. T.
Validating the Optimization of a Building Occupancy Monitoring Software System.
DOI: 10.5220/0013559400003964
In *Proceedings of the 20th International Conference on Software Technologies (ICSOFT 2025)*, pages 155-166
ISBN: 978-989-758-757-3; ISSN: 2184-2833

the art BOMS, Office Master 3000, to reduce the data samples required and the underlying energy consumption, for both data gathering and processing, while maintaining high accuracy for the occupancy state estimation. The resulting optimized BOMS relies on an intelligent knowledge-driven occupancy estimation so that it proactively identifies which high-level symbolic facts are needed to change the current state into an actual event. These facts are run through a semantically well-defined occupancy model to identify which *minimal combination* of non-intrusive sensors can supply data to confirm or contradict these facts, rather than blindly collect massive data from all sensors post-hoc to recognize ongoing activities. This enables to reduce the sensors frequency, and potentially switching sensors off for considerable time durations.

To validate the proposed optimization, we have modeled the optimized BOM in UPPAAL and used model checking to verify that the original non-functional properties (correctness, response time, performance and energy consumption) of the Office Master BOMS are still maintained post-optimization. Furthermore, the optimized BOMS is implemented in C++ where we analyzed the underlying performance and energy, and compare it to the original Office Master BOMS and to other state of the art BOMS algorithms.

The rest of the paper is structured as follows: Section 2 presents a background about BOMS and UPPAAL. Section 3 cites relevant related work. Section 4 presents the mathematical modeling and optimization of the proposed BOMS. Section 5 describes the specification and verification using UPPAAL. Implementation analysis and validation results are discussed in Section 6. Finally, Section 7 concludes the paper.

2 BACKGROUND

This section presents the background related to occupancy analysis, optimization and UPPAAL environment for formal verification and validation.

2.1 Occupancy State Estimation

Occupancy monitoring amounts to observe a specific area within a building to detect human presence and recognize occupancy activities (Ahmad et al., 2021). This typically entails gathering data from sensors that capture various features associated with human presence, such as breathing, movement and noise. The facts obtained from the sensors are then combined through an inference process to estimate the occupancy state (Yang et al., 2016).

For building management systems (BOMS) to effectively utilize this knowledge, real-time data is essential. This allows for timely actions to be taken based on the current occupancy state, thereby minimizing resource wastage and enhancing indoor comfort (Seghezzi et al., 2021; Sun et al., 2020).

With recent advancements enabling the deployment of BOMS on embedded devices using battery-powered sensors, optimizing the computational costs and energy consumption has become crucial. While reducing data sampling can serve as a means of optimization, it may compromise the accuracy of occupancy estimation (Yang et al., 2016). Hence, it is imperative to conduct post-optimization validation to ensure BOMS systems consistency and reliability (Boudjadar and Tomko, 2022).

2.2 BOMS Optimization

A BOMS optimization is the process of enhancing different performance metrics such as response time, resource utilization, and energy consumption (Minoli et al., 2017). The optimization can be achieved through a variety of techniques, including heuristics, mathematical solvers, and machine learning algorithms. However, it is crucial to ensure that improving one performance metric does not significantly degrade others or violate functional and non-functional properties (Boudjadar and Khooban, 2020; Boudjadar and Tomko, 2022). In the context of buildings, the need for optimization arises from their multifunctional nature, where individual spaces may serve different purposes. Therefore, tailoring a building space to its current activity is essential. For BOMS, optimization aims to reduce energy consumption and computational costs associated with data gathering and processing. The key objective of our optimization is to achieve real-time knowledge of the occupancy state, enabling dynamic adjustment of sensor frequencies based on the required data to confirm or deny an activity (Rault et al., 2014).

2.3 UPPAAL Model Checking

Model checking (E. M. Clarke et al., 1999; Baier and Katoen, 2008) is a widely used automated verification technique for examining complex reactive systems, including hardware components, embedded controllers, and network protocols. The method works by expressing the specifications of a system behavior in a semantically well-defined language and formalize the properties to look for in logical formu-

las such as LTL or CTL (Baier and Katoen, 2008), and then check whether the system behavior satisfies the expected qualities. We use UPPAAL (Boudjadar et al., 2013) for the post-optimization validation due to its expressiveness and ability to verify real-time systems.

UPPAAL is based on Timed Automata theory (Alur and Dill, 1994). As such, each UPPAAL template (timed automaton) is characterized by the following:

- A finite set of *locations*, with one designated as the *initial* location;
- A finite set of *transitions* connecting locations;
- A finite set of *variables*;
- A finite set of *actions* performed on the variables when executing a transition;
- A finite set of predicates, called *guards* allowing the execution of a transition:
- A finite set of *synchronization channels*.

A sequence of transitions represents the evolution of the specification. Each transition may be labeled with one or more actions, a guard and a synchronization. The *state* of a specification is given by the current location and the actual valuation of the variables. A specification in UPPAAL relies on combining multiple templates (network of timed automata) to create complex and modular systems. The composition can be synchronous where two transitions, one from each timed automaton, labeled with the same synchronization channel can be executed simultaneously.

3 RELATED WORK

Buildings occupancy monitoring has been thoroughly explored in the recent years where different monitoring models, analysis processes and optimizations are proposed (Lou et al., 2017; Rai et al., 2015; McKenna et al., 2015; Dai et al., 2020; Ahmad et al., 2021; Zhang et al., 2022; Jiang et al., 2022). The ultimate goal is to deliver highly accurate occupancy state estimation while optimizing the resources to deploy in order to achieve that mission such as energy consumption, accuracy and response time (Lasla et al., 2019; Trivedi and Badarla, 2020).

Lasla *et al* (Lasla et al., 2019) proposed an energy-efficient monitoring system for buildings using infrared sensors. The energy optimization comes from the optimal placing of sensors identified through a design exploration formulated as mixed integer linear problem. However, according to the authors, the accuracy of these experiments is dependent on the amount of data to be collected. The authors of (Nienaber et al., 2020) proposed the validation and tuning of different occupancy detection algorithms for multi-person offices where different abstraction models are considered, and the underlying impact of the decision accuracy is analyzed. Improvements of the algorithms have been conducted to achieve a trade-off between the granularity of the occupancy model, the assumptions to make and the decision accuracy. However, relying only on the CO_2 emission sensors has been proven to be less accurate as it depends on the physical state of the occupants. The authors of (Lou et al., 2017; Rai et al., 2015) proposed an agent-based occupancy analysis, that amounts to track the individual occupants rather than measuring the impact made by occupants on their surroundings. Although this alternative can enable accurate occupancy of the individuals, it can suffer from the scalability and the computation cost.

A state-based occupancy model has been proposed in (McKenna et al., 2015). It consists in creating stochastic data in terms of how probable an occupancy state will occur. This process enables to infer the location of occupants and their activities. However, the bottleneck can be related to the massive data gathering which drains sensor batteries and computation resources. Similarly, the authors of (Jiang et al., 2022) proposed a semantic (trust-based) model for occupancy analysis where the sensors triggering order and data are fed to a sequence matching scheme to estimate the occupancy state. The sequence matching scheme relies on known human activity sequences, which may lead to uncertainty when partial states or unknown scenarios are encountered.

Machine learning-based occupancy monitoring has also been explored. The authors of (Tien et al., 2022) proposed an occupancy monitoring model using convolutional neural networks and cameras to sample the building spaces. Although this approach achieves high accuracy, it requires massive data and may suffer from computation cost and privacy concerns.

Alternatively, a knowledge-based occupancy analysis has been proposed in (Ahmad et al., 2021). It consists in designing a domain-specific model encoding the activities to be carried out in the building and run the actual data through such a model to identify the occupancy state. In (Cala et al., 2015), the authors presented the evaluation and validation of an occupancy detection system based on the concentration of CO_2 in the indoor air. The validation amounted to check the similarity between the proposed BOMS outputs and the actual occupancy state reported by the occupants.

In this paper, we propose a knowledge-driven proactive BOMS system to analyze the actual occupancy state and identify the minimal data to collect in order to infer the ongoing activity in real-time, as an optimization of the actual Office Master 3000 BOMS. We analyze the proposed BOMS post-optimization to examine the performance improvement and confirm that the original functional and non-functional properties are maintained.

4 OPTIMIZED OCCUPANCY MONITORING

This section presents the proposed occupancy analysis model and optimization.

4.1 Occupancy State Specification

The original BOMS we consider monitor building spaces (classrooms, meeting rooms, etc) using CO_2 sensors, motion sensors, noise sensors and light sensors. The sensors are installed in different room ends, where each location contains a sensor from each category. The non-intrusive sensors are quantitative: light sensors in *lux*, CO_2 in parts per million (*ppm*), audio in decibels (*dB*), and motion in four levels: none (0), low (1), medium (2), and active (3).

Following the building functionality and sensors, we define the following language grammar $\mathcal{L}$ allowing to describe the different occupancy activities.

$$\begin{aligned} \mathcal{L} &\triangleq E_1|E_2|\ldots|E_n \\ E &\triangleq (F,c)|(F,c)\wedge E \\ F &\triangleq S|S\wedge F \\ S &\triangleq v\in[a,b] \\ c &\triangleq H|M|L \end{aligned}$$

The language specifies each potential occupancy event E through a set of features F, each of which is recognizable through a subset of sensors S if the value v of each sensor is within certain range [*a*,*b*]. The confidence level c states how much a feature is able to identify an event and can have values high (H), medium (M) or low (L). One can see that different events can have different features, which in turn can have different numbers of sensors associated.

We define a semantics function $||e|| : \mathcal{E} \rightarrow \mathcal{P}(\mathcal{S}) \times \mathcal{P}(c)$ to infer for each event the sensors and confidence levels from the event features, according to $\mathcal{L}$. $\mathcal{E}$ is the set of events, whereas $\mathcal{S}$ represents the set of sensors and $\mathcal{P}()$ is the power set (set of all subsets). Thanks to language $\mathcal{L}$ describing all potential events and features in a structured manner, we will be able to automate a thorough analysis of the occupancy state in real-time. This in fact will be done by projecting the sensor readings to identify the features present at the given time point. The features in turn enable to recognize the candidate events. An example of a feature is "*one or few people present in the front of classroom*". This feature will be identified if the values of sensor S_i and S_j are within these respective ranges: $v_{S_i} \in [410,700ppm] \wedge v_{S_j} \in [2,3]$. Accordingly, we specify the lecture event E_1 as follows:

$$\begin{aligned} E_1 \triangleq\ & \langle F_1,H\rangle \wedge \langle F_2,H\rangle \wedge \langle F_4,M\rangle \wedge \\ & \langle F_7,M\rangle \wedge \langle F_{10},L\rangle \wedge \langle F_{18},L\rangle \end{aligned}$$

The confidence level indicates how critical a sensor can contribute in recognizing an event. Nominally, the sensors of features having low confidence for a given event e can be discarded while e is being confirmed through high confidence features. The calculation and optimization of sensors frequency will be defined in the next section.

We define a valuation function V[S_i](t) to return the value of a given sensor S_i at time point t. A state of the proposed BOMS $\mathtt{s}$ at time t is given by the readings (V[S_i](t)) from each of the sensors sampled at time t.

$$\mathtt{s}(t) = \langle V[S_1](t), V[S_2](t), \ldots, V[S_n](t)\rangle$$

For the sake of notation, we use $\mathtt{s}$ to refer to the actual state, i.e. having the latest sensor readings. V returns the value of any sensor for any time point.

4.2 Knowledge-Driven Optimized Occupancy Analysis

The proposed BOMS optimization amounts to infer the sensors to use in order to recognize the expected event ε.The expected event can initially be obtained from the schedule of the monitored space. Alternatively, it can be set through a brute force data sampling by activating all sensors with the highest frequency, then the event recognized will be set to be the expected one. The optimization algorithm changes the expected event dynamically at runtime following the features recognized through occupancy analysis, as one can see in Algorithm 1. The sensors are planned on-the-fly where a sliding of the frequencies is calculated and applied following the expected event and the actual occupancy state.

Sensing Plan. A sensing plan R specifies the sensors and the frequency for each to sample occupancy data. Namely, R is given by $R = \langle (S_x,r_x),(S_y,r_y),(S_z,r_z),\ldots\rangle$, where S_i are sensors

and r_i are frequencies. For the sake of notation, we omit to specify time points as part of the sensing plans given that the knowledge expected for the occupancy state defines what sensors to operate and how often rather than the actual values of sensors. We simply write R to refer to the actual sensing plan.

Given that a sensor can be utilized to recognize different features of a given event, such a sensor runs the same (highest) frequency for the different features of the event but with respective reading ranges for each of the features.

Our occupancy analysis is carried out by combining the sensor readings and comparing the inferred occupancy state to the features of the expected event, or any other event of the language $\mathcal{L}$, in case the expected event cannot be approved. This results in 3 different cases: 1) good matching; 2) no matching; 3) partial matching.

Good Matching. This corresponds to the case when the actual occupancy state (sensor readings) matches a single event e. This results in updating ε to e, where a sensing plan is computed as follows:

$$R = \langle (S_1, r_1), .., (S_n, r_n) \mid \quad \forall i \; \exists c_j \; (S_i, c_j) \in ||e|| \wedge \; r_i = \begin{cases} 2 & \text{if } c_j = H \\ 4 & \text{if } c_j = M \\ 8 & \text{if } c_j = L \end{cases} \rangle$$

For the frequency calculation of each sensor, we associate frequencies of 2, 4 and 8 minutes respectively for sensors having high (H), medium (M) and low (L) confidence levels respectively. Those frequencies are analyzed empirically and proven to be sufficient to capture the occupancy features using the underlying confidence levels. Hence, an event being confirmed through few consecutive sampling iterations leads to an optimization of the sensing plan where sensors with the lowest confidence are discarded from the sensing plan as follows:

$$R = R / \{(S_i, r_i) \mid (S_i, r_i) \in ||e|| \wedge r_i = 8\}$$

Discarding a sensor from the sensing plan amounts to operating the sensor with the lowest frequency or switching off the sensor if it has already been operating at the lowest frequency for many consecutive iterations. Further elaboration on the optimization of sensing plans is provided in Section 4.3.

No Matching. No-match situation emerges when the outcomes from the actual sensing plan does not match any event of the language $\mathcal{L}$. To resolve such a problem, the sensing plan is expanded to include all sensors with high frequency (2 minutes) to be able to get a starting point so that either a good matching or a partial matching of an event occurs.

$$R = \langle (S_1, 2), (S_2, 2), \ldots, (S_n, 2) \rangle$$

In fact, the occurrence of a *no-match* case can be interpreted as a rejection or disapproval of the expected event ε.

Partial Matching. This case occurs when the actual occupancy state partially matches one or more events. To calibrate the sensing plan efficiently, we need to distinguish how many events partially match the actual state.

In case of a partial matching of the actual state to a *single event* e, we compute a new sensing plan as an update of the actual sensing plan R where all the sensors belonging to the features with high confidence in event e are included. Such sensors will run with highest frequency since they are given high confidence H. The new sensing plan is calculated as follows:

$$R = R \cup \{(S_i, r_i) | (S_i, r_i) \in ||e|| \wedge r_i = 2\}$$

In case this is insufficient to conclusively confirm the event that partially matches the actual state, the sensing plan will undergo an upgrade by incorporating sensors associated with features that exhibit medium confidence in event e. These sensors will operate at a medium frequency of 4 minutes, corresponding to their assigned medium confidence level M.

$$R = R \cup \{(S_i, r_i) | (S_i, r_i) \in ||e|| \wedge r_i = 4\}$$

In case the partial matching persists, the sensing plan will be expanded further to include all the sensor, being Off, belonging to the features with low confidence in event e. This will lead to either confirm the actual event e (*good matching* case), or disapprove it (*no-match* case). Accordingly, the update of the sensing plan R is made as follows:

$$R = R \cup \{(S_i, r_i) | (S_i, r_i) \in ||e|| \wedge r_i = 8\}$$

When facing partial matches to multiple events, it becomes crucial to pinpoint the ongoing event without engaging all sensors. To accomplish this, we begin by determining the semantic differences between the candidate events. For instance, if two events e_k and e_l exhibit partial alignment with the current state, we compute the set of distinguishing features between them, denoted as $D = ||e_k|| - ||e_l||$. The sensing plan is then updated as follows: $R = R \cup D$. This enables to decide which of the events is actually ongoing.

4.3 Sensing Plans Optimization

Our BOMS optimization is achieved through minimizing the real-time sensing plan to involve less sensors in tracking the actual occupancy state. It consists of tracking the actual occupancy state and matches it to the events defined in $\mathcal{L}$, so that our BOMS always ends with a good match case. Following the actual state, the optimization algorithm alternates between the different cases (good, partial, and no-match) with the lowest number of sensors active.

A sketch of the optimization algorithm is shown in Algorithm. 1. The optimization results from the case an event occurrence is consistently confirmed across consecutive iterations, resulting in a reduction in the number of sensors required to track the event.

Upon initial identification (good match) of an event, no optimization takes place. However, upon the second consecutive confirmation, all sensors assigned to features with low confidence are deactivated.

Similarly, with further consecutive confirmations of the expected event, sensors assigned to features with medium confidence are deactivated. Beyond this point, no further optimization occurs as the sensing plan comprises only a few high-confidence sensors that are sufficient to track and identify the ongoing occupancy event.

Algorithm 1: Controller Strategy.

1: $E_{Expected} \in Events$ ▷ Select an event to be monitored.
2: $AdjustSensors(E_{Expected})$ ▷ Sensors for the event.
3: $Monitor(E_{Expected})$ ▷ Monitor the event.
4: **switch** $MonitorStatus(E_{Expected})$: ▷ Status of event.
5: **case** *eventOccurred*: ▷ The event has occurred.
6: $OptimizeSensors(E_{Expected})$
7: Go to step 3 **break.**
8: **case** *otherEventOccurred*: ▷ Other event has occurred.
9: $E_{Expected} \leftarrow E_{Other}$ ▷ Select other event.
10: $AddSensors(E_{Expected})$ ▷ Add sensors for the event.
11: Go to step 3. **break.**
12: **case** *eventMaybeOccurred*: ▷ Event may have occurred.
13: Go to step 2. **break.** ▷ Monitor the event again.
14: **case** default: ▷ Unexpected event.
15: $TurnOnSensors()$ ▷ Gather more information using more sensors.
16: Go to step 3. **break.** ▷ Monitor the system.

5 FORMAL VERIFICATION AND VALIDATION

To conduct formal verification and validation of the proposed optimization, we modeled our BOMS in UPPAAL and specified the properties to look for in CTL logic. The model specification is modular and consists of three components (templates), a *Controller*, a *Sensor* and an *Environment*. The templates can be instantiated with different parameters to create different system architectures and configurations, and integrate new sensors, events and environments. The model is available here.

5.1 Model of the BOMS Control

The Controller process monitors the environment using Sensor instances, while the Environment simulates the occupancy events. The Controller template is illustrated in Figure 1, and summarized functionality of the Controller is depicted in Algorithm 1. The Controller recognizes occupancy events while minimizing the energy consumption of the Sensors. This is achieved by activating and deactivating the Sensor instances according to the occupancy current state using the *turn_on* and *turn_off* channels, respectively (which correspond to the functions *AddSensors()* and *OptimizeSensors()* in the algorithm) and by adjusting the sampling rate of the Sensors using the global variable *sample_frequency* (which corresponds to *AdjustSensors()* in the algorithm) since the energy consumption of a sensor depends on its sampling rate and the time duration it is active. The Controller only monitors one event at a time ($E_{Expected}$) and switches between the different events to monitor based on the occupancy state. In Algorithm 1, line 5 corresponds to a good matching to the expected event, line 8 describes no matching to the expected event but another event is recognized. Line 12 is a partial matching of the expected event, while line 12 describes no matching to any of the events in $\mathcal{L}$.

5.2 Sensors Modeling

The Sensor template, depicted in Figure 2, comprises two locations: *On* and *Off*. It samples the Environment and reports the sampled value to the Controller when active. The sensor value is read from the global array *environment_state*, that contains the current state of the Environment and is updated by the Environment template, through function *read()* which corresponds to $Monitor()$ in the algorithm. The periodic sensing, according to the frequency $sensor\_frequency[id]$, is modeled using a

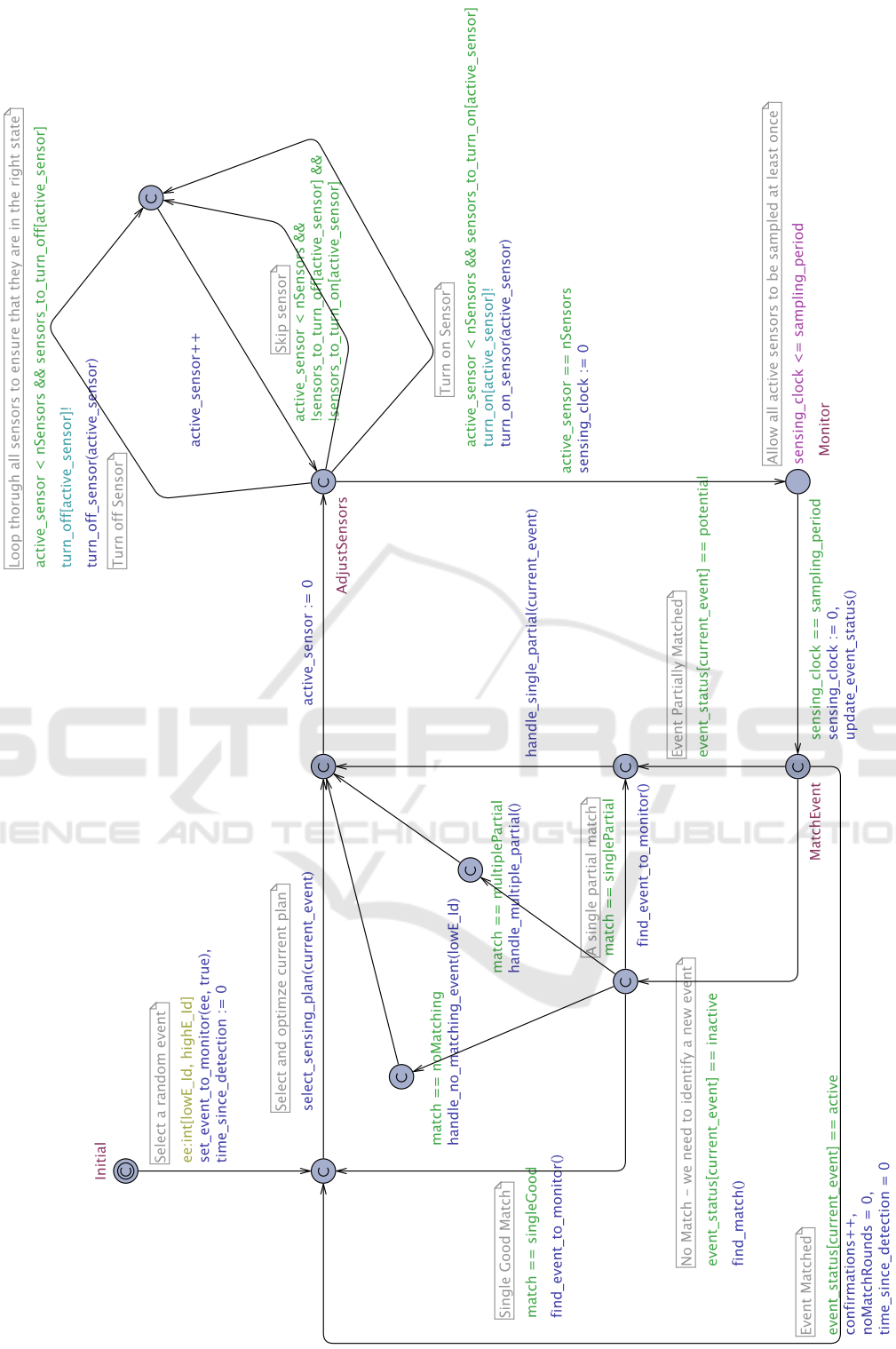

Figure 1: The Controller template.

state invariant in the *On* location and a clock constraint on transition performing the sampling based on the sampling rate of the Sensor (frequency) and its local clock variable *sample_clock*. The Controller

adjusts the sampling frequency by updating the array *sensor_frequency*[*id*], with which each sensor synchronizes using its Id. Furthermore, each sensor keeps track of its energy consumption by accounting for the number of samples taken and the number of time units spent in the location *On* through a global variable *energy_consumption*.

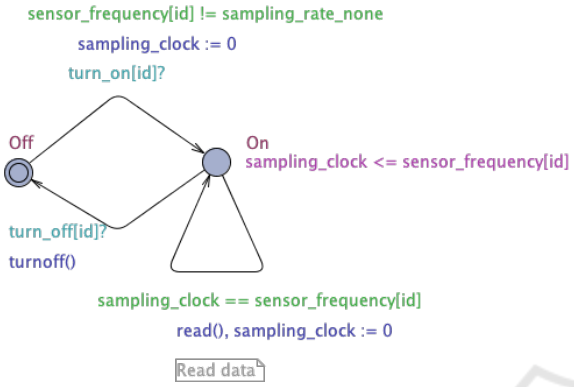

Figure 2: The Sensor template.

5.3 Environment Modeling

The Environment template, depicted in Figure 3, comprises two locations and periodically generates occupancy events. The events are generated non-deterministically upon which a global variable representing the current occupancy state is updated. Such a variable is read by the function *read()* of the different sensors. The periodic behavior of the Environment is modeled using a clock constraint on the transition that generates the new events based on the global variable *event_period* that determines the time between two events. The *event_period* has to be much larger than *sampling_period* to ensure that the Controller has time to detect the event. The Environment is made finite by constraining the number of events generated before the Environment goes into an *Off* state.

5.4 Validation Properties

We specified and verified the following properties for the optimized BOMS using UPPAAL. The queries are given in the formulas 1, 2, 3, 4, 5 and also available here:

- *Correctness:* The Controller should be able to recognize all the events generated by Environment.
- *Energy Consumption:* The energy consumption of the system should be less than certain threshold.
- *Response Time:* The response time of the system, defined as the time it takes for the Controller to detect an event, should be less than certain threshold.
- *Resilience:* The system should exhibit resilience, ensuring that the Controller is always able to devise a strategy to monitor the occupancy, even if the system is in an unexpected state.
- *Accuracy:* Each event recognized by the Controller must match the corresponding event generated by the Environment.
- *Performance:* The total number of accumulated sensor samples is always less than or equal to certain threshold.

Such properties are specified as follows:

$$\text{Correctness} \triangleq \text{Env.Execute} \rightarrow (\text{Ctrl.event} = \text{Env.event}) \quad (1)$$

$$\text{Energy} \triangleq \text{A}\Box\ \text{energyConsumption} \leq \text{E}_{\text{max}} \quad (2)$$

$$\text{Response time} \triangleq \text{A}\Box\ \text{ctrl.timeSinceDetection} \leq \text{R}_{\text{max}} \quad (3)$$

$$\text{Resilience} \triangleq \text{A}\Box\ \text{not deadlock} \quad (4)$$

$$\text{Performance} \triangleq \text{A}\Box\ \text{active\_sensor} \leq \text{X} \quad (5)$$

For the readers who are not familiar with the UPPAAL syntax, it should be noted that $p \rightarrow q$ corresponds to $A[](p => A <> q)$ in CTL.

The response time of the system is monitored using the clock variable *timeSinceDetection* that is reset every time the Controller detects an event. The response time property ensures that the Controller will always detect an event within certain time period (3 sampling periods in our case: 2, 4 and 8 minutes). The resilience property ensure liveness of the optimized BOMS system, by which the Controller can always derive a strategy and evolve following the occupancy state. The energy consumption property is simply the sum of the energy consumption of each Sensor, and is verified by ensuring that the energy consumption of the system is less than a certain threshold obtained from the proactivation algorithm. Finally, the correctness property is verified to ensure that any event that occurs in the BOMS will eventually be detected by the Controller.

Our formal verification demonstrates that the aforementioned properties of the original state of the art BOMS are satisfied by our proactivation BOMS, that is an optimization of the Office Master 3000 BOMS.

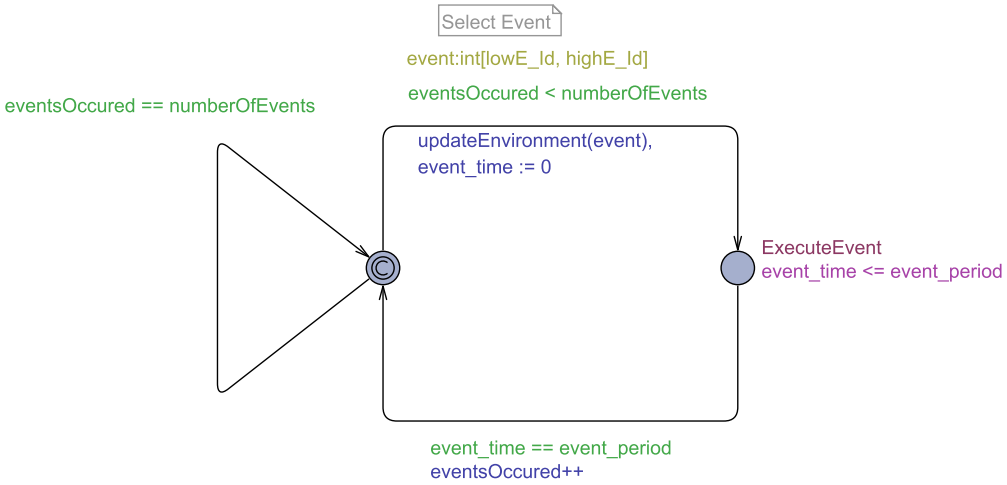

Figure 3: The Environment template.

6 IMPLEMENTATION AND EXPERIMENTAL VALIDATION

The proposed BOMS system and optimized occupancy tracking algorithm have been implemented in C++. The implementation is made modular as an integration of the three components (sensors, environments and controllers) so that new sensors, events and environments can be integrated and monitored in a straightforward manner. For reproducibility, the software implementation and data are available here.

We carried out an experiment of a 12-hour period to evaluate the performance and energy saving of our optimized BOMS system. To this end, we considered only one environment (building space) together with the corresponding controller and 2 hubs of sensors each of which is composed of 4 sensors (CO_2, motion, noise, light).

The key metrics we considered in the evaluation are the decision accuracy, the percentage of spared sensor samples and the energy saved due to having sensors turned off and/or operating with low frequency. The number of sensors being active at runtime is depicted in Figure 4. One can see that the sensing plan of our optimized BOMS rarely operates all of the eight sensors thanks to the knowledge-driven analysis. The number of active sensors fluctuates from 3 to 6 in the first 500 minutes due to different features and occupancy states detected. After 500 minutes, the monitored building space is mostly empty where the only event recognized is `Empty`, this is why the number of active sensors is reduced and fluctuates between 2 and 4.

As a result, the accumulated number of sensor samples collected by our optimized BOMS for 12 hours is **4212**. The total number of samples needed for the original Office Master 3000 BOMS, where all the sensors run regularly with the same 2-minutes frequency would be *6480*. Thus, our BOMS enables to save *35%* of the data sampling compared to the Office Master 3000 BOMS and also the state to the art algorithms (Ortiz Perez et al., 2018; Abraham and Li, 2014) where sensors are permanently active, while achieving the same accuracy and state knowledge.

As for energy consumption, for a 12-hours operation our optimized BOMS enables a saving of *9.75kw*, (6480-4212)*0.0043 (Vafamand et al., 2020), compared to the original Office Master 300s that would need *27.8kw*, which represents a saving of **35%**.

The accuracy analysis is depicted in Figure 5. It refers to the matching of the occupancy events identified by our BOMS algorithm and the actual occupancy events. The achieved accuracy fluctuates mostly between 40% and 100% with an average mean of **84%**. This means that in 84% of the experiment time, the expected event (computed according to the optimized sensing plan) matches the actual event, whereas no-matching cases (lowest value for blue line) represent 3.7% of the experiment time, and partial-matching pattern is 12.3%.

Furthermore, one can see that there is a correlation of the accuracy analysis with the optimized sensing plan. The lower the number of sensors and the higher the state dynamics are, the lower the accuracy is. However, upon each low accuracy a new sensing plan is computed by integrating more sensor samples incrementally to solve the partial-matching and no-matching cases. This led to shorten the time periods

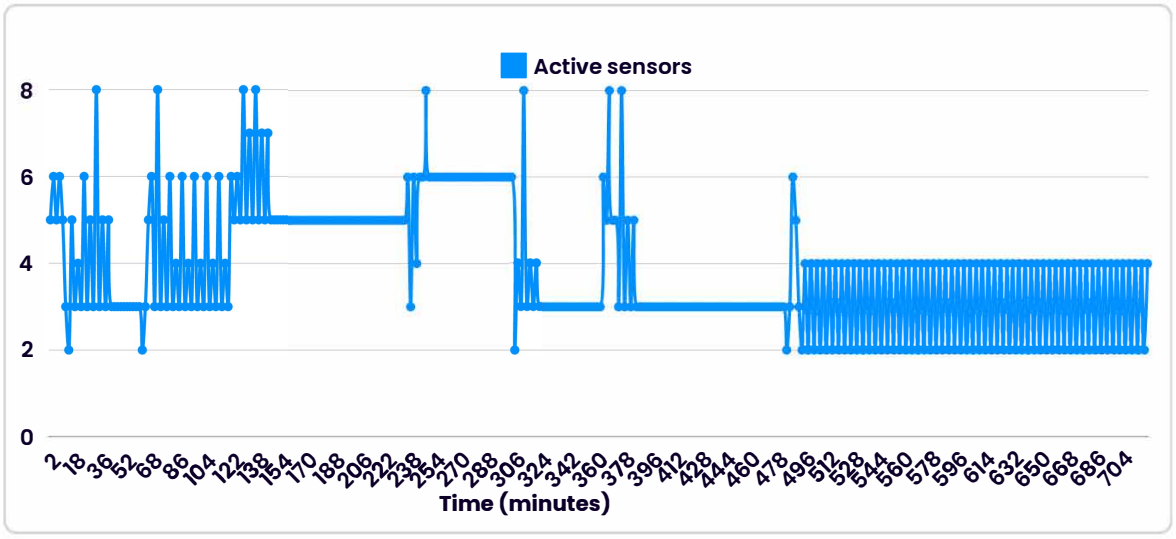

Figure 4: Number of sensors active at runtime.

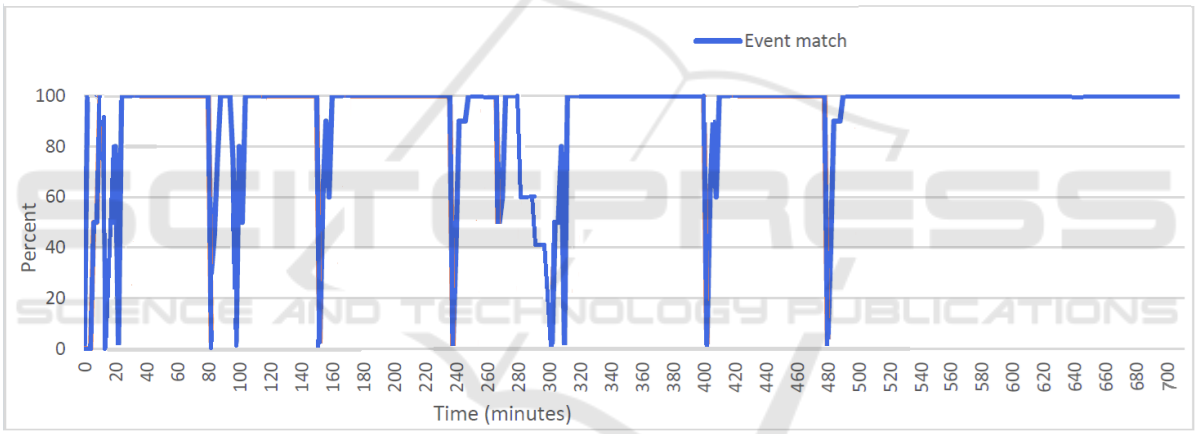

Figure 5: Accuracy of the optimized sensing plan.

for each accuracy drop. In average, each accuracy drop does not last more than 4-6 minutes.

A partial matching case is depicted in Figure 6 where the occupancy state recognized matches partially the features from 3 different events. As an example to solve this inconclusive situation would be to operate much more sensors to confirm/deny one of the features so that to exclude either the first 2 options (Lecture, Meeting) or the last option (Exercise session).

The outcomes resulting from the software experiment match the verification results conducted on the model using UPPAAL such as the maximum duration to recognize an event and the accumulated number of sensors for the same data set is always below 4300.

7 CONCLUSION

This paper proposed an intelligent knowledge-driven real-time occupancy monitoring solution (BOMS) as an optimization of the actual Office Master 3000 BOMS. The proposed analysis enables tuning the frequency of the sensors on-the-fly to reduce data sampling and energy consumption. Rather than collecting large amounts of sensor data to perform occupancy analysis post hoc; we proactively identify the minimal data relevant to the actual state following the semantics of the expected activities.

The proposed BOMS is mechanized in UPPAAL to perform formal verification and validation of the non-functional properties of the original BOMS post-optimization. An early proof-of-concept prototype has been implemented in C++. Furthermore, our so-

```
Event "Lecture" , matching partial, with features:
-#1: 1 or 2 people in front, not matched
-#2: Lecturing, not matched
-#3: Smartboard on, not matched
-#4: Many students, matched
-#5: Students quiet, not matched
-#6: Back lights off, not matched

Event "Meeting" , matching partial, with features:
-#1: Few people, matched
-#2: People talking, matched
-#3: Lights on, not matched
-#4: Person presenting, not matched
-#5: Smartboard on, not matched

Event "Exercise" , matching partial, with features:
-#1: No lecturing, not matched
-#2: Many people in room talking, not matched
-#3: Smartboard off, not matched
-#4: TA helping, matched
-#5: Lights on, not matched
```

Figure 6: Partial matching to multiple events.

lution has been tested and compared to baseline occupancy analysis. The experiment results showed that, while achieving a considerable reduction in computation cost (up to 35%) and energy consumption (up to 31%), it maintains high accuracy for the occupancy tracking (up to 84%).

As future work, we plan to integrate much more features and events to achieve a real-world occupancy state catalog. Moreover, we will also conduct a thorough analysis and validation of the proactivation on different case studies and compare the optimization results to those of machine-learning techniques.

REFERENCES

Abraham, S. and Li, X. (2014). A cost-effective wireless sensor network system for indoor air quality monitoring applications. *Procedia Computer Science*, 14(34).

Ahmad, J., Larijani, H., Emmanuel, R., Mannion, M., and Javed, A. (2021). Occupancy detection in non-residential buildings – a survey and novel privacy preserved occupancy monitoring solution. *Applied Computing and Informatics*, 17(2).

Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2).

Austin, M., Delgoshaei, P., Coelho, M., and Heidarinejad, M. (2020). Architecting smart city digital twins: Combined semantic model and machine learning approach. *Journal of Management in Engineering*, 36(4).

Azimi, S. and O'Brien, W. (2022). Fit-for-purpose: Measuring occupancy to support commercial building operations: A review. *Building and Environment Journal*, 22(212).

Baerentzen, M. U., Boudjadar, J., ul Islam, S., and Schultz, C. P. L. (2023). A knowledge-based proactive intelligent system for buildings occupancy monitoring. In *ICSOFT*, pages 680–687.

Baier, C. and Katoen, J.-P. (2008). *Principles of Model Checking*. The MIT Press.

Boudjadar, A., Vaandrager, F., Bodeveix, J.-P., and Filali, M. (2013). Extending uppaal for the modeling and verification of dynamic real-time systems. In Arbab, F. and Sirjani, M., editors, *Fundamentals of Software Engineering*. Springer Berlin Heidelberg.

Boudjadar, J., David, A., Kim, J. H., Larsen, K. G., Nyman, U., and Skou, A. (2014). Schedulability and energy efficiency for multi-core hierarchical scheduling systems. In *International Embedded Real-time Systems Sysmposium ERTS2*.

Boudjadar, J. and Khooban, M. (2020). A safety-driven cost optimization for the real-time operation of a hybrid energy system. In *Proceedings of the 27th International Conference on Systems Engineering (ICSEng)*.

Boudjadar, J. and Tomko, M. (2022). A digital twin setup for safety-aware optimization of a cyber-physical system. In *Proceedings of the 19th International Conference on Informatics in Control, Automation and Robotics*.

Cala, D., Matthes, P., Huchtemann, K., Streblow, R., and Müller, D. (2015). Co_2 based occupancy detection algorithm: Experimental analysis and validation for office and residential buildings. *Building and Environment Journal*, 86.

Costenaro, D. and Duer, A. (2012). The megawatts behind your megabytes: Going from data-center to desktop. In *ACEEE Summer Study on Energy Efficiency in Buildings*.

Dai, X., Liu, J., and Zhang, X. (2020). A review of studies applying machine learning models to predict occupancy and window-opening behaviours in smart buildings. *Energy and Buildings*, 20(223).

E. M. Clarke, J., Grumberg, O., and Peled, D. A. (1999). *Model Checking*. MIT Press.

Elkhoukhi, H., NaitMalek, Y., Berouine, A., Bakhouya, M., Elouadghiri, D., and Essaaidi, M. (2018). Towards a real-time occupancy detection approach for smart buildings. *Procedia Computer Science*, 18(134).

Jabirullah, M., Khan, A., Ali, M., Wajih, S., and Hussain, M. (2021). Iot-based occupancy monitoring techniques for energy efficient smart buildings. *Turkish Online Journal of Qualitative Inquiry*, 21(12).

Jiang, J., Wang, C., Roth, T., and Nguyen, C. (2022). Residential house occupancy detection: Trust-based scheme using economic and privacy-aware sensors. *IEEE Internet of Things Journal*, 9(3).

Jiang, W. and Yin, Z. (2015). Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM International Conference on Multimedia*.

Lasla, N., Doudou, M., Djenouri, D., Ouadjaout, A., and Zizoua, C. (2019). Wireless energy efficient occupancy-monitoring system for smart buildings. *Pervasive and Mobile Computing Journal*, 19(59).

Lou, X., Lam, K. P., Chen, Y., and Hong, T. (2017). Performance evaluation of an agent-based occupancy simu-

lation model. Technical report, School of Architecture, Carnegie Mellon University.

McKenna, E., Krawczynski, M., and Thomson, M. (2015). Four-state domestic building occupancy model for energy demand simulations. *Energy and Buildings*, 15(96).

Minoli, D., Sohraby, K., and Occhiogrosso, B. (2017). Iot considerations, requirements, and architectures for smart buildings - energy optimization and next-generation building management systems. *IEEE Internet of Things Journal*, 4(1).

Nienaber, F., Wolf, S., Wesseling, M., Calì, D., Muller, D., and Madsen, H. (2020). Validation, optimisation and comparison of carbon dioxide-based occupancy estimation algorithms. *Indoor and Built Environment Journal*, 29(6).

Ortiz Perez, A., Bierer, B., Scholz, L., Wollenstein, J., and Palzer, S. (2018). A wireless gas sensor network to monitor indoor environmental quality in schools. *Sensors Journal*, 18(12).

Pan, S., Bonde, A., Jing, J., Zhang, L., Zhang, P., and Noh, H. Y. (2014). Boes: Building occupancy estimation system using sparse ambient vibration monitoring. In *Proceedings of Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems SPIE*.

Rai, S., Wang, M., and Hu, X. (2015). A graph-based agent-oriented model for building occupancy simulation. In *Proceedings of the Symposium on Agent-Directed Simulation*.

Rault, T., Bouabdallah, A., and Challal, Y. (2014). Energy efficiency in wireless sensor networks: A top-down survey. *Computer Networks*, 14(67).

Salimi, S. and Hammad, A. (2019). Critical review and research roadmap of office building energy management based on occupancy monitoring. *Energy and Buildings*, 19(182).

Salimi, S. and Hammad, A. (2020). Sensitivity analysis of probabilistic occupancy prediction model using big data. *Building and Environment*, 20(172).

Seghezzi, E., Locatelli, M., Pellegrini, L., Pattini, G., Di Giuda, G. M., Tagliabue, L. C., and Boella, G. (2021). Towards an occupancy-oriented digital twin for facility management: Test campaign and sensors assessment. *Applied Sciences*, 11(7).

Shokrollahi, A., Persson, J. A., Malekian, R., Sarkheyli-Hägele, A., and Karlsson, F. (2024). Passive infrared sensor-based occupancy monitoring in smart buildings: A review of methodologies and machine learning approaches. *Sensors Journal*, 24(5).

Sun, K., Zhao, Q., and Zou, J. (2020). A review of building occupancy measurement systems. *Energy and Buildings*, 20(216).

Teng, S. Y., Touš, M., Leong, W. D., How, B. S., Lam, H. L., and Masa, V. (2021). Recent advances on industrial data-driven energy savings: Digital twins and infrastructures. *Renewable and Sustainable Energy Reviews*, 21(135).

Tien, P. W., Wei, S., Calautit, J. K., Darkwa, J., and Wood, C. (2022). Real-time monitoring of occupancy activities and window opening within buildings using an integrated deep learning-based approach for reducing energy demand. *Applied Energy*, 22(308).

Trivedi, D. and Badarla, V. (2020). Occupancy detection systems for indoor environments: A survey of approaches and methods. *Indoor and Built Environment Journal*, 29(8).

Vafamand, N., Boudjadar, J., and Khooban, M. H. (2020). Model predictive energy management in hybrid ferry grids. *Energy Reports*, 6:550–557.

Yang, J., Santamouris, M., and Lee, S. L. (2016). Review of occupancy sensing systems and occupancy modeling methodologies for the application in institutional buildings. *Energy and Buildings*, 16(121).

Zhang, H.-B., Zhang, Y.-X., Zhong, B., Lei, Q., Yang, L., Du, J.-X., and Chen, D.-S. (2019). A comprehensive survey of vision-based human action recognition methods. *Sensors Journal*, 19(5).

Zhang, W., Wu, Y., and Calautit, J. K. (2022). A review on occupancy prediction through machine learning for enhancing energy efficiency, air quality and thermal comfort in the built environment. *Renewable and Sustainable Energy Reviews*, 22(167).