International Journal on Advances in Systems and Measurements

2025 vol. 18 nr. 1&2

The *International Journal on Advances in Systems and Measurements* is published by IARIA.
ISSN: 1942-261x
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Systems and Measurements
Volume 18, Number 1 & 2, 2025

International Journal on Advances in Systems and Measurements
Volume 18, Numbers 1 & 2, 2025

CONTENTS

A Note on a Syntactical Measure of the Time-Space Complexity of Stack Programs

Emanuele Covino
Dipartimento di Informatica
Universitá degli Studi di Bari
Bari, Italy
emanuele.covino@uniba.it

***Abstract*—We introduce a programming language operating on stacks and a syntactical measure σ, such that a natural number $\sigma(\mathsf{P})$ is assigned to each program P. The measure considers how the presence of loops defined over a size-increasing (and non-size-increasing) subprogram influences the complexity of the program itself. Functions computed by a program of σ-measure n are exactly those computable by a Turing machine with running time in $\mathcal{E}^{n+2}$ (the $n+2$-th Grzegorczyk class). Programs of σ-measure 0 compute the polynomial-time computable functions. Thus, we have a syntactical characterization of functions belonging to the Grzegorczyk hierarchy; this result represents an improvement with respect to previous similar results. We then extend this approach to the definition of programs with simultaneous time and space bounds in the same hierarchy.**

***Index Terms*—Polynomial-Time Complexity; Grzegorczyk Hierarchy; Imperative Programming Languages; Stack Programs.**

I. Introduction

In this paper, we expand our earlier work [1] on the definition of a programming language operating on stacks, and a syntactical measure σ, such that functions computed by a program of σ-measure n are exactly those computable by a Turing machine with running time in the $n+2$-th Grzegorczyk class $\mathcal{E}^{n+2}$.

The definition of a class of functions with a given complexity is usually provided by introducing an explicit bound on time and/or space resources used by a Turing Machine during the computation of the functions. Other approaches capture complexity classes by means of some form of *limited recursion*; the first characterization of this type has been given by Cobham [2], who shows that the polynomial-time computable functions are exactly those that are definable by *bounded recursion on notation*, starting from a set of simple basic functions. In the recent years, a number of characterizations of complexity classes has been given, showing that they can be captured by means of various forms of *ramified recursion*, without any explicitly bounded scheme of recursion. Initiated by Simmons [3], Bellantoni and Cook [4] and Leivant [5] - [6], one can find functional characterization of linear-space/time computable functions LINSPACE and LOGSPACE [7], polynomial time [8], polynomial space [9] [10], the elementary functions [10] [11], non-size-increasing computations [12], among the others.

A different approach can be found in [13] [14] [15] [16]; more recently, in [17] [18]. The properties of *imperative* programs (such as complexity, resource utilization, termination) are now investigated by analyzing their syntax only. In particular, the properties of a programming language operating on stacks are studied in [15]; the language supports loops over stacks, conditionals and concatenation, besides the usual pop and push operations (see Section II for the detailed semantics). The natural concept of μ-measure is then introduced; it is a syntactical method by which one is able to assign to each program P a number $\mu(\mathsf{P})$. It is proved the following *bounding theorem*: functions computed by stack programs of μ measure n have a length bound $b \in \mathcal{E}^{n+2}$ (the $n+2$-th Grzegorczyk class), that is $|f(\vec{w})| \leq b(|\vec{w}|)$; as a consequence, stack programs of measure 0 have polynomial running time, and programs of measure n compute functions whose time complexity is in the $n+2$-th finite level of the Grzegorczyk hierarchy. This result provides a measure of the impact of nesting loops on computational complexity; if a stack Z is updated into a loop controlled by a stack Y and, afterwards, Y is updated into a loop controlled by Z, we have a so called *top circle* in the program; when this circular reference occurs into an external loop, a blow up in the complexity of the program is produced. The μ-measure is a syntactical way to detect top circles; each time one of them occurs in the body of a loop, the μ measure is increased by 1 (see below, Section III and definition 3.1).

There are various ways of improving the measure μ (for instance, see [16]), since it is an undecidable problem whether or not a function computed by a given stack program lies in a given complexity class. In this paper, we provide a refinement of μ, starting from the following consideration: a program whose structure leads the μ-measure to be equal to n contains n nested top circles, and this implies, by the bounding theorem, that the program has a length bound $b \in \mathcal{E}^{n+2}$. Suppose now that some of the sequences of pop and push (or, in general, some of the subprograms) iterated into the main program leave unchanged the overall space used; since not increasing programs can be iterated without

leading to any growth in space, the effective space bound is lower than the bound obtained via the μ-measure, and it can be evaluated by an alternative measure σ. While μ grows each time a top circle appears in the body of a loop, σ grows only for *increasing* top circles. In other words, the new measure does not consider those situations in which some (potentially harmful) operations are performed, but their overall balance is negative. We prove a new bounding theorem using the σ-measure, providing a more appropriate bound to the complexity of stacks programs.

Starting from this result, we present in this note a slight modification of the stack programs, the *non-space-increasing* programs. The only general requirement is that each not-increasing program never adds a digit to a stack without erasing another one (or more) from the same or from another stack; thus, a run of a program cannot exceed the overall length of the registers. Together with the σ-measure, this restriction allows us to characterize the set of functions which are computable by a Turing machine with time and space bounds simultaneously imposed. Indeed, we define the class of functions $\mathcal{B}^{m,n}$, that is the class of functions computable by a stack program P;Q (a run of P followed by a run of Q), with $\sigma(\mathsf{P}) = m$, $\sigma(\mathsf{Q}) = n$, and Q a not-increasing stack program. We prove that each function in $\mathcal{B}^{m,n}$ can be simulated by a Turing machine with running time in $\mathcal{E}^{n+2}$, and space not exceeding a bound in $\mathcal{E}^{m+2}$; conversely, each Turing machine with running time bounded by a function in $\mathcal{E}^{n+2}$ and, simultaneously, with space bounded by a function in $\mathcal{E}^{m+2}$ can be simulated by a stack program in $\mathcal{B}^{m,n}$. This result represents an extension to the space complexity case of Kristiansen and Niggl's result (following the problem raised in [12] and [19]), and a generalization to the finite levels of the Grzegorczyk hierarchy of our [11]. A sensible sequel of this note should be the definition of another measure, depending on σ, that should provide a way to evaluate the space complexity of the whole set of stack programs.

In Section II, we recall concepts and definitions of stack programs and of the Grzegorczyk hierarchy. In Section III, we recall the definition of μ-measure. In Section IV, we introduce the definition of the new σ-measure and the new bounding theorem. In Section V, we introduce the definition of the time/space classes and the related bounding theorems. Conclusions and future work can be found in Section VI.

II. Preliminaries on the Grzegorczyk hierarchy and on stack programs

In this section, we recall the definition of the Grzegorczyk hierarchy, and fundamentals on stack programs and their computations; the reader is referred to [15] [19] [20] [21] for complete definitions and properties.

Definition 2.1: Given a unary function f, the k-th *iterate* of f (denoted with f^k) is $f^0(x) = x$ and $f^{k+1}(x) = f(f^k(x))$.

Definition 2.2: The *principal functions* $E_1, E_2, E_3, \ldots$ are $E_1(x) = x^2 + 2$ and $E_{n+2}(x) = E^x_{n+1}(2)$ (the x-th iterate of E_{n+1}).

Definition 2.3: A function f is defined by *bounded recursion* from functions g, h, and b if for all $\vec{x}$, y one has $f(\vec{x}, 0) = g(\vec{x})$, $f(\vec{x}, y) = h(\vec{x}, y, f(\vec{x}))$, and $f(\vec{x}, y) \leq g(\vec{x}, y)$.

Definition 2.4: The n-th Grzegorczyk class $\mathcal{E}^n$ is the least class of functions containing the initial functions zero, successor, projections, maximum and E_{n-1}, and closed under composition and bounded recursion.

Stack programs operate on variables serving as stacks; they contain arbitrary words over a fixed alphabet Σ, and they are manipulated by running a program built from imperatives push(a,X), pop(X), and nil(X) combined by sequencing, conditional, and loop statements (respectively, P;Q, if top(X)≡a then [P], and foreach X [P]).

Definition 2.5: The operational semantics of stack programs are defined as follows:

1) push(a,X) pushes a letter a on the top of the stack X;
2) pop(X) removes the top of X, if any; it leaves X unchanged, otherwise;
3) nil(X) empties the stack X;
4) if top(X)≡a [P] executes P if the top of the stack X is equal to a;
5) $\mathsf{P}_1;\ldots;\mathsf{P}_k$ are executed from left to right;
6) foreach X [P] executes P for $|\mathsf{X}|$ times

with the restriction that no imperatives over X may occur in the body of a loop foreach X [P] (i.e., in P), and that the loop is executed call-by-value; X is the *control stack* of the loop. $|\mathsf{X}|$ stands for the length of the word stored in X.

The notation $\{A\}\mathsf{P}\{B\}$ means that if the condition expressed by the sentence A holds before the execution of P, then the condition expressed by the sentence B holds after the execution of P.

Definition 2.6: A stack program P *computes* a function $f : (\Sigma^*)^n \to \Sigma^*$ if P has an output variable O and n input variables $\bar{\mathsf{X}} = \mathsf{X}_{i_1}, \ldots, \mathsf{X}_{i_n}$ among stacks $\mathsf{X}_1, \ldots, \mathsf{X}_m$ such that $\{\bar{\mathsf{X}} = \vec{w}\}\mathsf{P}\{\mathsf{O} = f(\vec{w})\}$, for all $\vec{w} = w_1, \ldots, w_n \in (\Sigma^*)^n$.

For a fixed program P, two sets of variables are defined: $\mathcal{U}(\mathsf{P}) = \{\mathsf{X} | \mathsf{P}$ contains an imperative push(a,X)$\}$ and $\mathcal{C}(\mathsf{P}) = \{\mathsf{X} | \mathsf{P}$ contains a loop foreach X [Q], and $\mathcal{U}(\mathsf{Q}) \neq \emptyset\}$. Variables in $\mathcal{U}(\mathsf{P})$ can be altered by a push during a run of P, while variables in $\mathcal{C}(\mathsf{P})$ control a loop occurring in P. The two sets are not disjoint.

Definition 2.7: X *controls* Y in the program P (denoted with $\mathsf{X} \prec_{\mathsf{P}} \mathsf{Y}$) if P contains a loop foreach X [Q], and $\mathsf{Y} \in \mathcal{U}(\mathsf{Q})$; the transitive closure of $\prec_{\mathsf{P}}$ is denoted by $\overset{\mathsf{P}}{\rightarrow}$.

Consider the following program:

P_1:= foreach X_1[... foreach X_l [push (a,Y)]]

If words $v_1 \ldots v_l, w$ are stored in $\mathsf{X}_1 \ldots \mathsf{X}_l$, Y, respectively, before P_1 is executed, then Y holds the word $w\mathsf{a}^{|v_1|\cdots|v_l|}$

after the execution of P_1. The depth of loop-nesting is a necessary condition for high computational complexity, but it is not a sufficient condition. Now, consider the following two programs:

```
P2:= nil(Y); push(a,Y); nil(Z); push(a,Z);
     foreach X [nil(Z); foreach Y [push(a,Z); push(a,Z)];
                nil(Y); foreach Z [push(a,Y)]]

P3:= nil(Y); push(a,Y); nil(Z);
     foreach X [
         foreach Y [push(a,Z); push(a,Z); push(a,Y)]]
```

Even if both P_2 and P_3 have nesting depth 2, if w is initially stored in X, then Z holds the word $a^{2^{|w|}}$ after P_2 is executed, while $a^{|w|(|w|+1)}$ is stored in Z after the execution of P_3. Thus, we see that P_3 runs in polynomial time, whereas P_2 has exponential running time. This happens because of the (control) circle contained inside the outermost loop in P_2: inside the loop governed by X, first Y *controls* Z (in that Z is updated via push(a,Z) inside a loop governed by Y), and then Z *controls* Y in the same sense. In contrast, there is no such circle in P_3. Stack programs where each body of a loop statement is circle-free compute exactly the functions computable within polynomial time, and must be separated from those programs with loops that cause a blow up in running time.

III. The μ-measure on stack programs

Starting from the previous relation $\xrightarrow{P}$, a measure over the set of stack programs is introduced in [15].

Definition 3.1: Let P be a stack program. The *μ-measure of* P (denoted with $\mu(P)$) is defined as follows, inductively:

1) $\mu(\mathsf{pop}) = \mu(\mathsf{push}) = \mu(\mathsf{nil}) := 0$;
2) $\mu(\mathsf{if\ top(X)}{\equiv}\mathsf{a\ [Q]}) := \mu(Q)$;
3) $\mu(P;Q) := \max(\mu(P), \mu(Q))$;
4) $\mu(\mathsf{foreach\ X\ [Q]}) := \mu(Q) + 1$, if Q is a sequence $Q_1; \ldots; Q_l$ with a *top circle* (that is, if there exists Q_i such that $\mu(Q_i) = \mu(Q)$, some Y controls some Z in Q_i, and Z controls Y in $Q_1; \ldots; Q_{i-1}; Q_{i+1}; \ldots; Q_l$); $\mu(\mathsf{foreach\ X\ [Q]}) := \mu(Q)$, otherwise.

To focus on the critical case where P is a loop foreach X [Q], assume that $\mu(Q)$ is already determined. Suppose that Q is a sequence $Q_1; \ldots; Q_l$, in which case $\mu(Q)$ is $\max(Q_1, \ldots, Q_l)$. Then a blow up in running time can only occur if Q has a top circle, that is, Q has a circle with respect to a control variable Y of some component Q_i of maximal μ-measure $\mu(Q)$. In this case, $\mu(P)$ is defined as $\mu(Q)$+1. In all other cases, $\mu(P)$ is defined as $\mu(Q)$. Given that all primitive instructions receive μ-measure 0, one easily verifies for the examples above that $\mu(P_1)$=$\mu(P_3)$=0, whereas $\mu(P_2)$=1.

The core of [15] is the following bounding theorem.

Lemma 3.1: Every function f computed by a stack program of μ-measure n has length bound $b \in \mathcal{E}^{n+2}$ satisfying $|f(\vec{w})| \leq b(|\vec{w}|)$, for all $\vec{w}$. In particular, if P computes a function f, and $\mu(P) = 0$, then f has a polynomial length bound, that is, there exists a polynomial p satisfying $|f(\vec{w})| \leq p(|\vec{w}|)$.

Let $\mathcal{L}^n_\mu$ be the class of all functions which can be computed by a stack program of μ-measure $n \geq 0$, and let $\mathcal{G}^n$ be the class of all functions which can be computed by a Turing machine in time $b(|\vec{w}|)$, for some $b \in \mathcal{E}^n$. As a consequence of the bounding lemma, the following result holds.

Theorem 3.1: For $n \geq 0$: $\mathcal{L}^n_\mu = \mathcal{G}^{n+2}$.

Proof. By mutual simulation. The "$\subseteq$" inclusion starts from an arbitrary stack program Q of μ-measure n. Each imperative program occurring in Q can be simulated on a Turing machine in time $q(|X|)$, with q a polynomial. Let $TIME_P(\vec{w})$ denote the number of steps in a run of P on $\vec{w}$, and let $P^\sharp$ be the result of replacing in P each imperative imp with imp;push(a,V), for a new variable V. Then the program TIME(P): $\equiv$ nil(V);$P^\sharp$ has μ measure n and is such that $\{\vec{X} = \vec{w}\}\mathsf{TIME(P)}\{|V| = TIME_P(\vec{w})\}$. By the bounding theorem, we obtain a length bound $b \in \mathcal{E}^{n+2}$ satisfying $\{\vec{X} = \vec{w}\}\mathsf{TIME(P)}\{|V| \leq b(|\vec{w}|)\}$. Hence, there exists a Turing machine which simulates P within time $q(b(|\vec{w}|)) \cdot b(|\vec{w}|)$.

The opposite "$\supseteq$" inclusion shows that a single-tape Turing machine running in time $b(|\vec{w}|)$ can be simulated by a stack program with μ-measure n, provided that $b \in \mathcal{E}^{n+2}$. The mentioned program has the following form:

```
P:≡  TIME-BOUND(Y);              μ-measure is n
     INITIALIZE(L,Z,R);          μ-measure is 0
     foreach Y [SIM-MOVES];      μ-measure is 0
     OUTPUT(R;O);                μ-measure is 0
```

In order to write the first of the four subprograms, observe that, for each positive n, one can find a sequence LE[n+1] such that $\mu(\mathsf{LE[n+1]}) = n$ and $\{Y = w\}\mathsf{LE[n+1]}\{|Y| = E_{n+1}(|w|)\}$ (that is, LE[n+1] computes the $n+1$-th function of the sequence of principal functions $E_1, E_2, \ldots$, for a given input w); indeed, for some new variable U, LE[n+1] can be defined by:

```
LE[n+2]:≡  nil(U); foreach Y [push(a,U)];
           nil(Y); push(a,Y); push(a;Y);
           foreach U [LE[n+1]]
```

Recalling that there exists a constant c such that $b(x) \leq E^c_{n+1}(x)$, we have

```
TIME-BOUND(Y):≡  nil(Y);
                 foreach X [push(a,Y)]
                 LE[n+1];...;LE[n+1]      (c times).
```

We have that $\{Y = w\}\mathsf{TIME\text{-}BOUND(Y)}\{X = w, |Y| = E^c_{n+1}(|w|)\}$. We omit the details about the definition of INITIALIZE (it sets the registers to the initial configuration of the Turing machine) and OUTPUT (it returns the result of the computation in a fixed register). The program SIM-MOVES is in the form $\mathsf{MOVE}_1;\ldots;\mathsf{MOVE}_k$, where MOVE_i simulates the i-th move of the Turing machine, operating on two stacks L and R (one for the left side of the tape, the other for the right side; see [15] for a detailed

description of this simulation).

IV. The σ-measure and a new bounding theorem

In the rest of the paper, we denote with imp(Y) an imperative pop(Y), push(a,Y), or nil(Y); we denote with mod($\bar{X}$) a *modifier*, that is a sequence of imperatives operating on the variables occurring in $\bar{X} = X_1, \ldots, X_n$. We introduce a modified definition of *circle*, which better matches our new measure.

Definition 4.1: Let Q be a sequence in the form $Q_1; \ldots; Q_l$. There is a *circle* in Q if there exists a sequence of variables $Z_1, Z_2, \ldots, Z_l$, and a permutation π of $\{1, \ldots, l\}$ such that $Z_1 \overset{Q_{\pi(1)}}{\rightarrow} Z_2 \overset{Q_{\pi(2)}}{\rightarrow} \ldots Z_l \overset{Q_{\pi(l)}}{\rightarrow} Z_1$. The subprograms $Q_1, \ldots, Q_l$ and the variables $Z_1, \ldots, Z_l$ are *involved* in the circle.

For sake of simplicity, we will consider $\pi(i) = i$, that is the case $Z_1 \overset{Q_1}{\rightarrow} Z_2 \overset{Q_2}{\rightarrow} \ldots Z_l \overset{Q_l}{\rightarrow} Z_1$; proofs and definitions holds in the general case too.

Definition 4.2: Let P be a stack program and let Y be a given variable. The *σ-measure of* P *with respect to* Y (denoted with $\sigma_Y(P)$) is defined as follows, inductively (with $sg(z) = 1$ if $z \geq 1$, $sg(z) = 0$ otherwise):

1) $\sigma_Y(\text{mod}(\bar{X})) := sg(\sum \hat{\sigma}_Y(\text{imp(Y)}))$, for each imp(Y) $\in$ mod($\bar{X}$), where
   $\hat{\sigma}_Y(\text{push(a,Y)}) := 1$;
   $\hat{\sigma}_Y(\text{pop(Y)}) := -1$;
   $\hat{\sigma}_Y(\text{nil(Y)}) := -\infty$;
   $\hat{\sigma}_Y(\text{imp(X)}) := 0$, with Y$\neq$X;
2) $\sigma_Y(\text{if top Z} \equiv \text{a[P]}) := \sigma_Y(P)$;
3) $\sigma_Y(P_1;P_2) := \max(\sigma_Y(P_1), \sigma_Y(P_2))$, with $P_1;P_2$ not a modifier;
4) $\sigma_Y(\text{foreach X [Q]}) := \sigma_Y(Q)+1$, if there exists a circle in Q, and a subprogram Q_i s.t.
   (a) Y and Q_i are involved in the circle;
   (b) $\sigma_Y(Q) = \sigma_Y(Q_i)$;
   (c) the circle is increasing;
   $\sigma_Y(\text{foreach X [Q]}) := \sigma_Y(Q)$, otherwise,

where a circle is *not increasing* if, denoted with $Q_1, Q_2, \ldots, Q_l$ and with $Z_1, Z_2, \ldots, Z_l$ the sequences of subprograms and, respectively, of variables involved in the circle, we have that $\sigma_{Z_i}(Q_j) = 0$, for each $i := 1 \ldots l$ and $j := 1 \ldots l$. If the previous condition does not hold, we say that the circle is *increasing*.

Note that the σ_Y-measure of a modifier (see (1) in the previous definition) is equal to 1 only when, in absence of nil's, the overall number of push's over Y is greater than the number of pop's over the same variable, that is, only when a growth in the length of Y is produced. Moreover, note that the "otherwise" case in (4) can be split in three different cases. First, there are no circles in which Y is involved. Second, Y is involved, together with a subprogram Q_i, in a circle in Q, but it happens that $\sigma_Y(Q_i)$ is lower than $\sigma_Y(Q)$ (this means that there is a blow-up in the complexity of Y in $\sigma_Y(Q_i)$, but this growth is still bounded by the complexity of Y in a different subprogram of Q). Third, Y is involved in some circles in Q, but each of them is not increasing (that is, according to the previous definition, each variable Z_i involved in each circle does not produce a growth in the complexity of the subprograms Q_j involved in the same circle). This implies that the space used during the execution of the external loop foreach X [Q] is basically the same used by Q (this is not a surprising fact: one can freely iterate a not increasing program without leading an harmful growth). In all three cases the σ-measure must remain unchanged: it is increased only when an increasing top circle occurs and when at least one of the variables involved in that circle causes a growth in the space complexity of the related subprogram, simultaneously (that is, if there exists a p such that $\sigma_{Z_p}(Q_p) > 0$).

In the following definition, we extend the measure to the whole set of variables occurring in a stack program.

Definition 4.3: Let P be a stack program. The *σ-measure of* P is $\sigma(P) := \tilde{\sigma}(P) \dot{-} 1$, where $\dot{-}$ is the usual cut-off subtraction, and

1) $\tilde{\sigma}(\text{mod}(\bar{X})) := 0$
2) $\tilde{\sigma}(\text{if top Z} \equiv \text{a [P]}) := \max(\sigma_Y(\text{if top Z} \equiv \text{a [P]}))$, for all Y occurring in P;
3) $\tilde{\sigma}(P_1;P_2) := \max(\sigma_Y(P_1;P_2))$, for all Y occurring in P, with $P_1;P_2$ not a modifier;
4) $\tilde{\sigma}(\text{foreach X [Q]}) := \max(\sigma_Y(\text{foreach X [Q]}))$, for all Y occurring in P.

Note that $\sigma(P) \leq \mu(P)$, for each stack program P. Note also that we are using the previously defined $\hat{\sigma}_Y$ to detect all the *increasing* modifiers, for a given variable Y (this is done setting $\hat{\sigma}_Y$ equal to 1); but, once detected, we don't have to consider them in the evaluation of the σ-measure. This is the reason of the "$\dot{-}1$" part in the previous definition.

In the rest of the paper we introduce a reduction procedure between stack programs, denoted with $\rightsquigarrow$, and we prove a new bounding theorem.

Definition 4.4: P and Q are *space equivalent* if $\{\bar{X} = \vec{w}\}P\{|\bar{X}| = m\}$ implies that $\{\bar{X} = \vec{w}\}Q\{|\bar{X}| = O(m)\}$. This is denoted with $P \approx_s Q$.

Definition 4.5: Let A be a stack program such that $\mu(A) = n+1$, and $\sigma(A) = m$, with $m < n+1$; the program $\rightsquigarrow A$ is obtained as follows:

1) if A is foreach X [R], with $\mu(R) = \sigma(R) = n$, and denoted with $C_1, \ldots, C_l$ the top circles in R, and with $A_{i1}, \ldots, A_{ip}$ the variables involved in C_i, for each i, we have that $\rightsquigarrow A$ is the result of changing each imp(A_{ij}) into nop(A_{ij}) (a *no-operation* imperative);
2) if A is foreach X [R], with $\mu(R) > \sigma(R)$, , we have that $\rightsquigarrow A$ is equal to foreach X [$\rightsquigarrow R$];
3) if A is $A_1;A_2$ and $\max(\mu(A_1), \mu(A_2)) = \mu(A_1)$, we have that $\rightsquigarrow A$ is equal to $\rightsquigarrow A_1;A_2$;
   simmetrically, if $\max(\mu(A_1), \mu(A_2)) = \mu(A_2)$, we have that $\rightsquigarrow A$ is equal to $A_1;\rightsquigarrow A_2$;

if $\mu(\mathsf{A}_1) = \mu(\mathsf{A}_2)$, we have that $\rightsquigarrow$A is equal to $\rightsquigarrow\mathsf{A}_1;\rightsquigarrow\mathsf{A}_2$;

4) if A is if top(X)≡a [R], we have that $\rightsquigarrow$A is equal to if top(X)≡a [$\rightsquigarrow$R].

Lemma 4.1: Given a stack program P, with $\mu(\mathsf{P}) = n+1$ and $\sigma(\mathsf{P}) = n$, there exists a stack program $\rightsquigarrow\mathsf{P}$ such that $\mu(\rightsquigarrow \mathsf{P}) = n$, $\sigma(\rightsquigarrow \mathsf{P}) = n$, and $\mathsf{P} \approx_s \rightsquigarrow\mathsf{P}$.

Proof. (by induction on n). Base. Let $\mu(\mathsf{P}) = 1$ and $\sigma(\mathsf{P}) = 0$. In the main case, P is in the form foreach X [Q], with a not-incresing circle occurring in Q. Applying $\rightsquigarrow$ to P, we obtain a program P′ whose σ-measure is still 0, and whose μ-measure is reduced to 0, because $\rightsquigarrow$ has broken off the circle in P that leads μ from 0 to 1 (i.e., in P′, there are no more push's on the variables involved in the circle). Note that P can decrease the length of the stacks involved in the circle, while P′ does not perform any operation in the same circle. Thus, P′ can increase its variables only by a linear factor; indeed, if $\{\bar{\mathsf{X}} = \vec{w}\}\mathsf{P}\{|\bar{\mathsf{X}}| = m\}$ we have that $\{\bar{\mathsf{X}} = \vec{w}\}\mathsf{P}'\{|\bar{\mathsf{X}}| = cm\}$, where c is a constant depending on the structure of P: thus, $\mathsf{P} \approx_s \mathsf{P}'$.

Step. Let $\mu(\mathsf{P}) = n+2$ and $\sigma(\mathsf{P}) = n+1$. Let P be in the form foreach X [Q], and let C be a top circle occurring in Q, with $\mu(\mathsf{Q}) = n+1$; we have two cases: (1) $\sigma(\mathsf{Q}) = n+1$, or (2) $\sigma(\mathsf{Q}) = n$.

(1) In this case C is a not-increasing circle, because it has been detected by μ, but not by σ. Applying $\rightsquigarrow$ to P, we obtain a program P′ such that $\sigma(\mathsf{P}') = n+1$, $\mu(\mathsf{P}') = n+1$, and $\mathsf{P} \approx_s \mathsf{P}'$.

(2) In this case C is an increasing circle, detected by μ and σ. We have that (by the inductive hypothesis) there exists a program Q′ such that $\mu(\mathsf{Q}') = n$, $\sigma(\mathsf{Q}') = n$, and $\mathsf{Q} \approx_s \mathsf{Q}'$. Starting from P, we build a new program P′=foreach X [Q'] . We have that $\mu(\mathsf{P}') = \mu(\mathsf{Q}')+1 = n+1$, $\sigma(\mathsf{P}') = \sigma(\mathsf{Q}')+1 = n+1$, and $\mathsf{P} \approx_s \mathsf{P}'$ as expected.

The cases $\mathsf{P}_1;\mathsf{P}_2;\ldots;\mathsf{P}_k$ and if top(X)≡a [P] can be proved in a similar way.

Theorem 4.1: Every function f computed by a stack program P such that $\mu(\mathsf{P}) = n$ and $\sigma(\mathsf{P}) = m$, with $n > m$, has a length bound $b \in \mathcal{E}^{m+2}$ satisfying $|f(\vec{w})| \leq b(|\vec{w}|)$.

Proof. Let k be $\mu(\mathsf{P}) - \sigma(\mathsf{P})$. Then by k applications of Lemma 4.1, we obtain a sequence $\mathsf{P} =: \mathsf{P}_0, \mathsf{P}_1, \ldots, \mathsf{P}_k$ of stack programs such that, for all $i < k$,

$$\mu(\mathsf{P}_{i+1}) = \mu(\mathsf{P}) - i,\ \sigma(\mathsf{P}_i) = \sigma(\mathsf{P}_{i+1}), \text{ and } \mathsf{P}_i \approx_s \mathsf{P}_{i+1}.$$

By Kristiansen and Niggl's bounding theorem, P_k has a length bound in $\mathcal{E}^{\sigma(\mathsf{P})+2}$, and so does P, by transitivity of $\approx_s$.

Let $\mathcal{L}^n_\sigma$ be the class of all functions that can be computed by a stack program of σ-measure $n \geq 0$, and let $\mathcal{G}^n$ be the class of all functions which can be computed by a Turing machine in time $b(|\vec{w}|)$, for some $b \in \mathcal{E}^n$. As a consequence of Theorem 4.1, and similarly to what has been recalled in Section III, the following result holds.

Theorem 4.2: For $n \geq 0$: $\mathcal{L}^n_\sigma = \mathcal{G}^{n+2}$.

V. Restrictions to time-space complexity

In this section, we introduce some syntactical restrictions to the stack programming language, and we prove that, when combined with the σ-measure, they allow us to evaluate the time and (simultaneously) the space complexity of a program written according to the new syntax. In particular, we define $\mathcal{B}^{m,n}$ as the class of functions computable by a stack program in the form P;Q (a run of P followed by a run of Q), where $\sigma(\mathsf{P}) = m$, $\sigma(\mathsf{Q}) = n$, and where Q is a *not-increasing* stack program (see below for the definition). As we promised in the introduction, we prove that each function in $\mathcal{B}^{m,n}$ can be simulated by a Turing machine with running time in $\mathcal{E}^{n+2}$, and space in $\mathcal{E}^{m+2}$, and, conversely, each Turing machine with running time bounded by a function in $\mathcal{E}^{n+2}$ and, simultaneously, with space bounded by a function in $\mathcal{E}^{m+2}$ can be simulated by stack programs in $\mathcal{B}^{m,n}$.

Definition 5.1:

1) *Given a modifier M and a stack X the* rate of growth *of M with respect to X is the difference between the number of push(a,X) and the number of pop(X) occurring in M.*
2) *A* non-space-increasing stack program *is built from modifiers by sequencing, conditional and loop statements, provided that the rate of growth of each modifier with respect to each stack is negative, or equal to* 0.

The following lemma shows that a not-increasing program cannot increase, as expected, the overall length of the registers on which it operates (except for a constant c which depends only on the structure of the modifiers occurring into the program).

Lemma 5.1: Let P be a not-increasing stack program; there exists a constant $c \geq 0$ such that

$$\{\vec{X} = \vec{w}\}P\{|\vec{X}| \leq |\vec{w}| + c\}.$$

Proof. Given the definition of not-increasing programs, it is natural to observe that they cannot add digits to their inputs; there is only one exception to this fact, that is when one or more of the stacks on which a program operates are empty (and thus, some of the pop's occurring into the program have no effect). In this case, only a constant number of digits can be added to some stack, hence the constant c of the theorem; in particular, $c = \sum_{\mathsf{X}} c_{\mathsf{X}}$, with c_{X} the maximum number of push(a,X) occurring into the modifiers of the program. For example, consider P:≡ pop(X);pop(X);push(a,X);push(a,X), with X equal to the empty word. In this case the first two pop have no effect on X, while the rest of the program pushes two a's into X, returning a value whose length is greater than the input's length, notwithstanding P is still not-increasing. If P occurs into a loop, the number of digits added at the end of the loop's run is still two, since each run of pop(X);pop(X) erases two digits from X, and each run of push(a,X);push(a,X) adds two digits to X.

The proof proceeds by induction on the structure of the program. The base case is obvious, given the definition of modifiers with negative rate of growth. As for the step, let P and Q two not-increasing programs, operating on $\vec{X}$. By the inductive hypothesis, $\{\vec{X} = \vec{w}\}P\{|\vec{X}| \leq |\vec{w}| + c_1\}$, and $\{\vec{X} = \vec{w}\}Q\{|\vec{X}| \leq |\vec{w}| + c_2\}$; the sequence P;Q is such that $\{\vec{X} = \vec{w}\}P;Q\{|\vec{X}| \leq |\vec{w}| + \max\{c_1, c_2\}\}$. The same happens for the conditional and the loop cases.

Definition 5.2: Let h and k be two natural numbers.

1) *$\mathcal{B}^{h,k}$ denotes the class of all functions computable by a stack program in the form P;Q, where $\sigma(P) = h$, $\sigma(Q) = k$, and Q is a non-space-increasing stack program.*
2) *If $k \leq h \leq k + 1$, $\mathcal{G}^{h,k}$ denotes the class of functions computable by a Turing machine on input $\vec{w}$ within time $t(|\vec{w}|)$ (with $t \in \mathcal{E}^h$), and in space $s(|\vec{w}|)$ (with $s \in \mathcal{E}^k$), simultaneously.*

The "=" in the following theorem stands for the mutual inclusion between two classes of functions. In this case, between the class of functions computed by our version of stack programs and the class of functions computable by Turing machines with given time and space bounds.

Theorem 5.1: For m and n two natural numbers ($m \leq n \leq m + 1$), we have $\mathcal{B}^{m,n} = \mathcal{G}^{n+2,m+2}$.

The result comes from the next two lemmas.

Lemma 5.2: For m and n two natural numbers ($m \leq n \leq m + 1$), we have $\mathcal{B}^{m,n} \subseteq \mathcal{G}^{n+2,m+2}$.

Proof. Let S be a stack program in the form P;Q, with $\mu(P) = m$, $\mu(Q) = n$, and Q a not-increasing program. Let $\text{TIME}_S(\vec{w})$ denote the number of steps in a run of the program S on $\vec{w}$ (a step is an execution of an imperative), and let $\text{SPACE}_S(\vec{w})$ denote the overall number of registers' cells used by S during a run.

By theorem 3.1 one obtains a Turing machine which simulates P on input $\vec{w}$ within time bounded by $b(|\vec{w}|)$, with $b \in \mathcal{E}^{m+2}$; hence, the space used by the Turing machine simulating P is bounded by $b(|\vec{w}|)$. Let $\vec{v}$ the sequence of variables such that $\{\vec{X} = \vec{w}\}P\{\vec{X} = \vec{v}\}$; Q runs on $\vec{v}$. By theorem 3.1 there exists a Turing machine simulating Q on $\vec{v}$, in time $b'(|\vec{v}|)$, with $b' \in \mathcal{E}^{n+2}$. The overall time is $\text{TIME}_P + \text{TIME}_Q \leq b(|\vec{w}|) + b'(|\vec{v}|) \leq b'(|\vec{v}|)$ (being $m \leq n$), with $b' \in \mathcal{E}^{n+2}$. As for the evaluation of the space complexity, note that Q is a not-incresing program, then it uses $\text{SPACE}_Q(\vec{v}) \leq |\vec{v}| + c$; hence, the overall space used by S is bounded by $b(|\vec{w}|)$, with $b \in \mathcal{E}^{m+2}$. The sequence of the two Turing machines gives us the desired result.

Lemma 5.3: For m and n two natural numbers ($m \leq n \leq m + 1$), we have $\mathcal{G}^{n+2,m+2} \subseteq \mathcal{B}^{m,n}$.

Proof. Let M be an arbitrary Turing machine belonging to $\mathcal{G}^{n+2,m+2}$, that is running (on input $\vec{w}$) in time $t(|\vec{w}|)$, with $t \in \mathcal{E}^{n+2}$, and in space $s(|\vec{w}|)$, with $s \in \mathcal{E}^{m+2}$. M can be simulated by two Turing machines, M_P and M_Q, respectively time-bounded by $s(|\vec{w}|)$ and $t(|\vec{v}|)$; M_P delimits (in $s(|\vec{w}|)$ steps) the space that M_Q will use during its run. Moreover, M_Q does not exceed its input.

By theorem 3.1 there exists a program P which simulates M_P, and such that $\sigma(P) = m$; for the same reason, there exists a program Q which simulates M_Q, with $\sigma(Q) = n$. We cannot use this program in our proof, since it is an increasing program; indeed, according to the second part of theorem 3.1, Q contains a subprogram LE[n+2] which stores into Y the number of times that the SIM-MOVES related to M_Q will be executed. We are looking for an alternative procedure to define, for each n and for each input x, the appropriate sequence of $E_n(x)$ SIM-MOVES needed in order to simulate M_Q correctly. This is done by following the definition of the program LE[n+2] in [15]; the intended output of our procedure is the appropriate number of sequenced SIM-MOVES.

$\mathcal{L}E_{(n+2)} :\equiv$ for each x do {u:=u+1; SIM-MOVES};
x:=2;
for each u do $\mathcal{L}E_{(n+1)}$.

The sequence of SIM-MOVES we obtain executing c times $\mathcal{L}E_{(n+2)}$ is a not-incresing stack program, since each SIM-MOVES never pushes a digit into a stack without popping another digit from the other one. The reader should note that $\mathcal{L}E$ is not a stack program, but its outputs are. Setting M $:\equiv$ SIM-MOVES; ...; SIM-MOVES ($E_n^c(x)$ times), and setting Q $:\equiv$ INITIALIZE;M;OUTPUT, we have that the stack program P;Q is in $\mathcal{B}^{m,n}$, by definition 5.2.

VI. Conclusions

We have defined a syntactical measure σ that considers how the iteration of imperative stack programs affects the complexity of the programs themselves. In particular, this measure only counts those loops in which programs with a size-increasing effect (w.r.t. the final length of the result) are iterated. Each time such a loop is built over other loops, the σ-measure is increased by 1. Other measures detect potentially harmful loops, but are not able to distinguish between size-increasing and non-size-increasing loops. It is undecidable to know if a function computed by a given stack program lies in a given complexity class, but our measure represents an improvement when compared to previously defined measures. We can assign a function computed by a stack program of σ-measure n to the $n+2-th$ Grzegorczyk class, and this class is lower in the hierarchy, when compared to the class obtained via other measures. We have extended this idea to the classification of programs that computes functions with simultaneous time and space bounds in the same hierarchy.

References

[1] E. Covino, "A Note on a Syntactical Measure of the Complexity of Programs," The Fifteenth International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking - COMPUTATION TOOLS 2024, April 14, 2024 to April 18, 2024 - Venice, Italy, ISBN: 978-1-68558-158-9, ISSN: 2308-4170.

[2] A. Cobham, "The intrinsic computational difficulty of functions," in Y. Bar-Hillel (ed), Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science, pp. 24-30, North-Holland, Amsterdam, 1962.
[3] H. Simmons, "The realm of primitive recursion," Arch. Math. Logic 27 (1988), pp. 177–188.
[4] S. Bellantoni and S. Cook, "A new recursion-theoretic characterization of the poly-time functions," Computational Complexity, no. 2, pp. 97-110, 1992.
[5] D. Leivant, "Subrecursion and lambda representation over free algebras," in S. Buss, P. Scott (Eds.), Feasible Mathematics, Perspectives in Computer Science, BirkhLauser, Boston, New York, 1990, pp. 281–291.
[6] D. Leivant, "Stratifed functional programs and computational complexity," in Conf. Record of the 20th Annual ACM Symposium on Principles of Programming Languages, New York, 1993, pp. 325–333.
[7] L. Kristiansen, "New recursion-theoretic characterizations of well known complexity classes," Fourth International Workshop on Implicit Computational Complexity (ICC'02), Copenhagen.
[8] D. Leivant, "Ramifed recurrence and computational complexity I: Word recurrence and poly-time," in P. Clote, J. Remmel (Eds.), Feasible Mathematics II, Perspectives in Computer Science, BirkhLauser, Basel, 1994, pp. 320–343.
[9] D. Leivant and J.-Y. Marion, "Ramified recurrence and computational complexity II: substitution and polyspace," in J. Tiuryn and L. Pocholsky (eds), Computer Science Logic, LNCS no. 933, pp. 486-500, 1995.
[10] I. Oitavem, "New recursive characterization of the elementary functions and the functions computable in polynomial space," Revista Matematica de la Universidad Complutense de Madrid, no. 10.1, pp. 109-125, 1997.
[11] E. Covino and G. Pani, "Diagonalization and the complexity of programs," The Ninth International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking, (COMPUTATION TOOLS 2018), February 18-22, 2018, Barcelona, Spain. ISBN: 978-1-61208-394-0. ISSN: 2308-4170
[12] M. Hofmann, "The strength of non-size-increasing computations," Principles of Programming Languages, POPL'02, Portland, Oregon, January 16-18th, 2002.
[13] N. Jones, "Program analysis for implicit computational complexity," Third International Workshop on Implicit Computational Complexity (ICC'01), Aarhus.
[14] N. Jones, "LOGSPACE and PTIME characterized by programming languages," Theoretical Computer Science, no. 228, pp. 151-174, 1999.
[15] L. Kristiansen and K.-H. Niggl, "On the computational complexity of imperative programming languages," Theoretical Computer Science, no. 318(1-2), pp. 139–161, 2004.
[16] L. Kristiansen and K.-H. Niggl, "The garland measure and computational complexity of imperative programs," Fifth International Workshop on Implicit Computational Complexity, (ICC '03), Ottawa.
[17] D. Leivant, "A generic imperative language for polynomial time," arXiv:1911.04026v2 [cs.LO], 2020.
[18] D. Leivant and J.-Y. Marion, "Primitive recursion in the abstract," Mathematical Structures in Computer Science, Cambridge University Press (CUP), 2020, 30 (1), pp. 33-43. 10.1017/S0960129519000112. hal-02573188.
[19] P. Clote, "Computation models and function algebra," in E. Grivor (Ed.), Handbook of Computability Theory, Elsevier, Amsterdam, 1996.
[20] H. E. Rose, Subrecursion: functions and hierarchies, Oxford University Press, Oxford, 1984.
[21] A. Grzegorczyk, "Some classes of recursive functions," Rozprawy Mat., Vol. IV, Warszawa, 1953.

Fuzzy Agent-Based Simulations of Cooperative Strategies for Task Allocation, Collision Avoidance, and Battery Charging Management of Autonomous Industrial Vehicles

Juliette Grosset
IMT Atlantique, IRISA, UMR 6074
Rennes, France
jgrosset10@gmail.com

Alain-Jérôme Fougères
ECAM Louis de Broglie, IT Laboratory
Rennes, France
alain-jerome.fougeres@ecam-rennes.fr

Moïse Djoko-Kouam
ECAM Louis de Broglie, IETR, UMR CNRS 6164
Rennes, France
moise.djoko-kouam@ecam-rennes.fr

Jean-Marie Bonnin
IMT Atlantique, IRISA, UMR 6074
Rennes, France
jm.bonnin@imt-atlantique.fr

***Abstract*—The paper presents a multi-agent simulation using fuzzy inference to explore the task allocation, collision avoidance, and battery charging management of mobile baggage conveyor robots in an airport, in an integrated way. The approach leverages V2X cooperation to enable real-time communication between mobile robots and airport infrastructure, enhancing adaptability thanks to a distributed system, adapting to variations in the availability of conveyor agents, their battery capacity, infrastructure resource availability, and awareness of the activity of the conveyor fleet. Dynamic factors, such as workload variations and communication between the conveyor agents and infrastructure are considered as heuristics, highlighting the importance of flexible and collaborative approaches in autonomous systems. The results highlight the effectiveness of adaptive fuzzy multi-agent models to optimize dynamic task allocation, adapt to the variation of baggage arrival flows, improve the overall operational efficiency of conveyor agents, and collision avoidance, and reduce their energy consumption through V2X-enabled cooperation.**

Keywords-autonomous industrial vehicle; dynamic task allocation; collision avoidance; V2X cooperation; fuzzy agent; agent-based simulation; Airport 4.0.

I. Introduction

This article significantly extends our previous conference paper [1], which initially introduced a fuzzy agent-based simulation of task allocation and battery charge management. In this extended version, we incorporate collision avoidance mechanisms, integrate V2X communication for real-time coordination with infrastructure, and enhance the multi-agent framework to support richer, more realistic scenarios with improved adaptability and operational efficiency. The deployment of Autonomous Industrial Vehicle (AIV) fleets in the context of Airport 4.0 raises several issues, all related to their real level of autonomy: acceptance by employees, vehicle localization, traffic flow, failure detection, collision avoidance and vehicle perception in changing environments. Simulation makes it possible to take into account the various constraints and requirements formulated by manufacturers and future users of these AIVs. Before starting to test AIV fleet traffic scenarios in often-complex airport situations, it is wise, if not essential, to simulate these scenarios [2]. Moreover, one of the main advantages of using simulations is that the results can be used without the need to apply a scaling factor. The main advantages of simulating mobile robot or AIV operations are: reducing the time and cost of developing an AIV, minimizing potential operational risks associated with AIVs, allowing to assessment of the feasibility of different AIV circulation scenarios at a strategic or operational level, allowing a rapid understanding of the operations carried out by AIVs, and identifying improvements in the layout configurations of the facilities hosting these AIVs [3]. Simulation also provides flexibility in terms of AIV deployment and allows studying the sharing of responsibility between the central server and the robots (local/global or centralized/decentralized balance) for the different operational decisions. Another advantage of simulations is to introduce humans into the scenarios in order to verify and validate, before the actual deployment of autonomous mobile robots, the safety of the coexistence and possible interactions between these AIVs and human operators [4]. Agent-based approaches are often proposed for the simulation of autonomous vehicles. They offer simulation contexts ranging from trajectory planning to optimal task allocation while allowing collision and obstacle avoidance [5]. Our current research focuses on the use of fuzzy agents to handle the levels of imprecision and uncertainty involved in modeling the behavior of simulated vehicles [6]. Indeed, fuzzy set theory is well suited to the processing of uncertain or imprecise information that must lead to decision-making by autonomous agents, used in activities such as the simulation of AIVs in an airport or product design [7]. Fuzzy agents can track the evolution of fuzzy information from their environment and from agents [8]. By interpreting the fuzzy information they receive or perceive, fuzzy agents interact within the multi-agent system of which they are a part. For example, a fuzzy agent can discriminate a fuzzy interaction value to assess its degree of affinity (or interest) with another fuzzy agent [9]. Thus, we develop a comprehensive study on utilizing fuzzy inference within multi-agent simulations to optimize task allocation and battery management for mobile baggage conveyor robots in airports. The proposed

simulation approach is designed to be highly adaptable, considering dynamic factors such as workload variations, battery capacities, and communication between agents and infrastructure. The results demonstrate that this adaptive fuzzy multi-agent model can significantly improve operational efficiency, adapt to variations in baggage arrival flows, and reduce energy consumption. This article is structured as follows: first in Section II, we present a state-of-the-art review of major concepts of fleets of AIV: task allocation, obstacle avoidance, battery recharging, V2X cooperation, and fuzzy agent-based simulation. Then, in Section III, we introduce a case study on fuzzy agent-based simulations of mobile baggage conveyors in an airport, where we present the simulation framework, the use of V2X cooperation, and task allocation strategies, both basic and fuzzy. We also integrate collision avoidance and speed adaptation into the simulations. In Section IV, we propose three improvements using fuzzy heuristics. Finally, we conclude on the proposed fuzzy dynamic task allocation strategies and present future research directions.

II. Major Concepts

A. Task Allocation

Task Allocation (TA) consists of optimally assigning a set of tasks to be performed by agents, actors, robots, or processes, grouped and organized within a cohesive system. This is the case for mobile multi-robot systems [10], AIV fleets [11], and applications in airports [12]. In the field of mobile robotics, the taxonomy presented in [13] has been defined to better characterize allocation and assignment functions to robots: Single Task for a Single Robot (STSR), Multiple Tasks for a Single Robot (MTSR), and Multiple Tasks for Multiple Robots (MTMR). These classifications enable tasks to be assigned to one or multiple robots, with various tasks being allocated to heterogeneous or multitasking robots. Moreover, De Ryck et al. [13] defined also: allocation modalities, such as instantaneous allocation or allocation extended in time. This last is linked to synchronization and precedent or time window constraints. As many combinations as exhaustively detailed by numerous surveys on the issue of multi-robot TA. Different solution models have been proposed for TA: based on optimization: exact algorithms, dynamic programming, (meta-)heuristics [10]; based on the Contract Net Protocol: inside an agent-based system, an initiating agent sends a call for proposals to all agents, chooses the best proposal received, and then informs all agents [11]; based on the concept of the market: announcement by an auctioneer, submission by bidders, selection by the auctioneer and award by the auctioneer [14]. Furthermore, different types of optimization objectives can be defined for this task allocation [13]: cost objectives (travel costs, such as time, distance, or fuel consumption), behavior objectives (ability of a robot to perform a task), reward objectives (payoff for performing a task), priority objectives (urgency to perform a task), and utility objectives (subtracting the cost from the reward or fitness). Task allocation and planning are often managed centrally, even semi-centrally when global and local planning are differentiated [15]. For the proper functioning of autonomous and dynamic systems, the requirements of flexibility, robustness and scalability, lead to consider decentralized mechanisms to react to unexpected situations. Autonomy and decentralization are two excessively linked notions to the extent that an autonomous system operates and makes decisions autonomously [16]. The problem of task allocation can also be thought of in a decentralized way [13]. For reasons of flexibility, robustness and scalability necessary in an Industry 4.0 or Airport 4.0 context, we are interested in decentralized task allocation solutions. These solutions, decomposed below, must be able to assign tasks to a fleet of robots. Particularly, solutions based on the market concept can easily be applied in a distributed context, where each mobile robot can become an auctioneer [17]. For each situation, a single mobile robot is appointed auctioneer and retains this role until the situation is definitively managed.

B. Obstacle Avoidance

Obstacle avoidance is a critical challenge in the deployment of AIV fleets, especially in dynamic and complex environments such as airports. It ensures safe and efficient navigation by preventing collisions with static and dynamic obstacles while maintaining operational efficiency. Currently, avoidance strategies are often implemented on a per-robot basis [18], without a coordinated collective approach. However, in the context of AIV fleets, a collective approach that incorporates multi-robot communication and coordination can significantly enhance adaptability and efficiency. Effective obstacle avoidance strategies must integrate three key components:

- Obstacle perception/detection: AIVs rely on onboard sensors (e.g., LiDAR, cameras, ultrasonic sensors) and perception algorithms to detect obstacles [19]. For this study, we assume that robots are already equipped with effective sensors and algorithms for detecting obstacles.
- Rerouting/trajectory planning: Once an obstacle is detected, AIVs must compute an alternative trajectory. While various rerouting methods exist, we focus on broader strategic decisions rather than specific algorithms of path planning or path finding [20, 21].
- Overall strategy decisions: Effective obstacle avoidance requires real-time decision-making mechanisms that adapt to both static and dynamic obstacles. This is the key focus of our study, as we are primarily interested in high-level decision-making mechanisms that enable effective obstacle avoidance in multi-robot systems. This includes multi-robot communication and coordination strategies, as well as real-time decision-making processes and algorithms.

Simulations play a crucial role in evaluating and optimizing obstacle avoidance strategies before real-world deployment.

C. Battery Recharging

Effective energy management is crucial for AIVs as it directly affects their operational efficiency and autonomy. Managing energy resources involves monitoring battery status, detecting technical anomalies, and performing necessary maintenance, as highlighted by [22]. Optimizing energy consumption requires a holistic approach that considers operational availability, energy efficiency [23], collaboration with dynamic infrastructure, and adaptation to changing conditions. Battery recharging in AIVs must balance individual and collective energy needs to maximize system efficiency. This balance is achieved through two key decision-making principles: 1. Local Equilibrium – Each AIV optimizes its own recharging schedule to maintain operational readiness. An AIV may initiate recharging when its battery level drops below a predefined threshold, ensuring it can complete assigned tasks without disruptions. 2. Global Equilibrium – This approach considers the energy demands of the entire fleet and the surrounding infrastructure [24]. Coordinating fleet-wide energy consumption prevents congestion at charging stations, reduces power spikes, and improves overall system efficiency. Strategies such as staggered recharging schedules, shared infrastructure utilization, and workload-based energy distribution help maintain global equilibrium. To ensure effective energy management, AIVs must strike a balance between these two levels: - Individual Decision-Making: Each AIV must autonomously determine its recharging needs based on real-time energy levels, workload, and immediate operational requirements [22]. This minimizes the risk of energy depletion while maintaining vehicle autonomy. - Collective Coordination: Simultaneously, AIVs must communicate and synchronize their charging needs with one another and with the infrastructure. This prevents bottlenecks caused by simultaneous charging demands and enhances overall system efficiency. A key objective in battery recharging is to optimize recharging cycles to minimize energy costs and avoid excessive power consumption during low-demand periods. Poor anticipation of energy needs can lead to inefficiencies and reduced system availability. Since AIV workloads fluctuate—alternating between high-activity and low-intensity phases—aligning energy consumption with operational demand ensures continuous and efficient performance. Reducing energy consumption is a major challenge for mobile robots, requiring optimization through well-defined cost functions. Power consumption is often modeled based on parameters such as speed and motor force [25, 26]. Various optimization techniques have been proposed such as:

- Genetic Algorithms – Methods such as those in [27] utilize genetic algorithms to minimize energy consumption through an optimal fuzzy logic controller.
- Fuzzy Logic Optimization – Mamdani fuzzy logic [28] optimizes speed profiles (trapezoidal/triangular) for both straight and curved paths to reduce power consumption.
- Pontryagin's Maximum Principle (PMP) – Applied in the railway sector [29], this principle optimizes train speed trajectories based on braking distance and can be adapted for AIV movement strategies.
- Power Integral Functions – These models refine AIV movement strategies to minimize overall energy usage by optimizing acceleration and deceleration patterns.

D. V2X Cooperation

To effectively complete assigned tasks, AIVs must coordinate, cooperate, and exchange information on environmental perceptions. This cooperation relies on Vehicle-to-Everything (V2X) communication, which encompasses three key communication modes:

1) Vehicle-to-Vehicle (V2V) Communication – Direct information exchange between AIVs enhances coordination and task execution efficiency. While extensively studied in the literature, specific applications include decentralized intersection traffic light synchronization [30], cooperative lane-change simulations [31], and traffic light optimization strategies [32]. V2V enables AIVs to share navigation data and obstacle detection information, facilitating adaptive decision-making.
2) Vehicle-to-Infrastructure (V2I) Communication – AIVs communicate with the surrounding infrastructure, such as smart warehouses, to receive real-time updates on environmental changes [33]. This mode enhances safety and efficiency by allowing AIVs to receive alerts about obstacles, traffic flow modifications, and operational constraints, thereby optimizing navigation and avoiding collisions.
3) Vehicle-to-Pedestrian (V2P) Communication – In environments shared with humans, AIVs leverage V2P communication to ensure safety and seamless human-robot collaboration [34]. This interaction is crucial when an AIV encounters obstacles requiring human intervention, as it enables efficient coordination between human operators and autonomous systems.

Collectively, these three communication modes form the V2X framework [35], enabling AIVs to operate efficiently in dynamic environments through real-time data exchange and cooperative decision-making. The European Telecommunication Standards Institute (ETSI) has established standardized communication protocols for Intelligent Transport Systems (ITS), which have been adapted for AIV cooperation [36]. Two key message types support autonomous decision-making and coordination:

- Decentralized Environmental Notification Messages (DENM) – Defined by ETSI EN 302 637-3 [37], these messages serve as alerts during unexpected events, allowing AIVs to broadcast real-time incident notifications within a specific geographic area.
- Cooperative Perception Messages (CPM) – Standardized in ETSI TR 103 562 [38], CPMs facilitate situational awareness by transmitting obstacle detection and

navigation updates. AIVs receiving CPMs can dynamically adjust their routes, preventing disruptions and optimizing task execution. Beyond these existing standards, emerging V2V communication approaches further enhance cooperation. For example, a multi-agent control strategy for connected urban buses [39] enables real-time movement adjustments based on downstream traffic conditions. By integrating enhanced V2X communication strategies, AIV fleets can achieve higher resilience, adaptability, and operational efficiency in dynamic environments.

E. Fuzzy Agent-Based Simulation

Many agent-based approaches are proposed for the simulation of autonomous vehicles. They offer simulation contexts ranging from trajectory planning [40] to optimal task allocation while allowing collision and obstacle avoidance [41]. Our current research focuses on the use of fuzzy agents to handle the levels of imprecision and uncertainty involved in modeling the behavior of simulated vehicles [6]. Fuzzy set theory is well suited to the processing of uncertain or imprecise information that must lead to decision-making by autonomous agents [7]. Most of the control tasks performed by autonomous mobile robots have been the subject of performance improvement studies using fuzzy logic [42]: navigation [43], obstacle avoidance [44], path planning [45], motion planning [46], localization of mobile robots [47], and intelligent management of energy consumption [48, 49]. An agent-based system is fuzzy if its agents have fuzzy behaviors or if the knowledge they use is fuzzy [50]. This means that agents can have: 1) fuzzy knowledge (fuzzy decision rules, fuzzy linguistic variables, and fuzzy linguistic values); 2) fuzzy behaviors (the behaviors adopted by agents because of fuzzy inferences); and 3) fuzzy interactions, organizations, or roles. Table I below proposes a model of fuzzy agents corresponding to the principles stated above.

TABLE I
FUZZY AGENT MODEL USED IN THE SIMULATIONS

$$\tilde{M}_\alpha = \langle \tilde{A}, \tilde{I}, \tilde{P}, \tilde{O} \rangle \quad (1)$$

$\tilde{\boldsymbol{A}}$ is a set of fuzzy agents; $\tilde{\boldsymbol{I}}$ is a set of fuzzy interactions between fuzzy agents; $\tilde{\boldsymbol{P}}$ is a set of fuzzy roles that fuzzy agents can perform; $\tilde{\boldsymbol{O}}$ is a set of fuzzy organizations defined for fuzzy agents (subsets of strongly linked fuzzy agents).

$$\widetilde{\alpha}_i = \langle \Phi_{\Pi(\widetilde{\alpha}_i)}, \Phi_{\Delta(\widetilde{\alpha}_i)}, \Phi_{\Gamma(\widetilde{\alpha}_i)}, K_{\widetilde{\alpha}_i} \rangle \quad (2)$$

$\boldsymbol{\Phi_{\Pi(\widetilde{\alpha}_i)}}$: is the $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$'s function of observation; $\boldsymbol{\Phi_{\Delta(\widetilde{\alpha}_i)}}$: is the $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$'s function of decision; $\boldsymbol{\Phi_{\Gamma(\widetilde{\alpha}_i)}}$: is the $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$'s function of action; $\boldsymbol{K_{\widetilde{\alpha}_i}}$: is the set of knowledge of the fuzzy agent $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$.

$$\Phi_{\Pi(\widetilde{\alpha}_i)} : (E_{\widetilde{\alpha}_i} \cup I_{\widetilde{\alpha}_i}) \times \Sigma_{\widetilde{\alpha}_i} \rightarrow \Pi_{\widetilde{\alpha}_i} \quad (3)$$

$$\Phi_{\Delta(\widetilde{\alpha}_i)} : \Pi_{\widetilde{\alpha}_i} \times \Sigma_{\widetilde{\alpha}_i} \rightarrow \Delta_{\widetilde{\alpha}_i} \quad (4)$$

$$\Phi_{\Gamma(\widetilde{\alpha}_i)} : \Delta_{\widetilde{\alpha}_i} \times \Sigma \rightarrow \Gamma_{\widetilde{\alpha}_i} \quad (5)$$

$\boldsymbol{E_{\widetilde{\alpha}_i}}$: is the $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$'s the set of fuzzy observed events; $I_{\widetilde{\alpha}_i}$: is the $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$'s set of fuzzy interactions; $\boldsymbol{\Sigma_{\widetilde{\alpha}_i}}$: is the $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$'s set of fuzzy states; $\boldsymbol{\Pi_{\widetilde{\alpha}_i}}$: is the $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$'s set of fuzzy perceptions; $\boldsymbol{\Delta_{\widetilde{\alpha}_i}}$: is the $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$'s set of fuzzy decisions; $\boldsymbol{\Gamma_{\widetilde{\alpha}_i}}$: is the $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}$'s set of fuzzy actions; $\boldsymbol{\Sigma}$: is the state of fuzzy agent-based system $\widetilde{\boldsymbol{M}}_{\boldsymbol{\alpha}}$.

$$\widetilde{l}_l = \langle \widetilde{\alpha}_s, \widetilde{\alpha}_r, \widetilde{\gamma}_c \rangle \quad (6)$$

$\widetilde{\boldsymbol{l}}_{\boldsymbol{l}}$: is a fuzzy interaction; $\widetilde{\alpha}_s$: is the fuzzy agent source of a fuzzy interaction; $\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{r}}$: is the fuzzy agent receiver of a fuzzy interaction; $\widetilde{\boldsymbol{\gamma}}_{\boldsymbol{c}}$: is a fuzzy communication act (*inform, diffuse, ask, reply, and confirm*, are used in the basic model).

III. CASE STUDY: FUZZY AGENT-BASED SIMULATION OF MOBILE BAGGAGE CONVEYORS IN AN AIRPORT

This case study proposes the simulation of mobile robots conveying baggage fleet in an airport (we will keep the name "AIV" for these conveyors). Fig. 1 shows the simulator's Human Computer Interface (HCI), which allows on the one hand, to visualize the arrival of baggage and the movements of 5 AIVs, and on the other hand, to follow the evolution of the different levels of indicators of the simulation (energy level, baggage level, charge level, and time level). The circulation scenario is detailed with a distance-oriented graph presented in Fig. 2.

Effective management of these AIVs requires an integrative approach that considers several factors, including the baggage arrival flow, the operational availability of the AIVs, their energy consumption, their communication, among themselves and with the infrastructure, and their adaptation to changing environmental conditions. In the case study, we analyze the

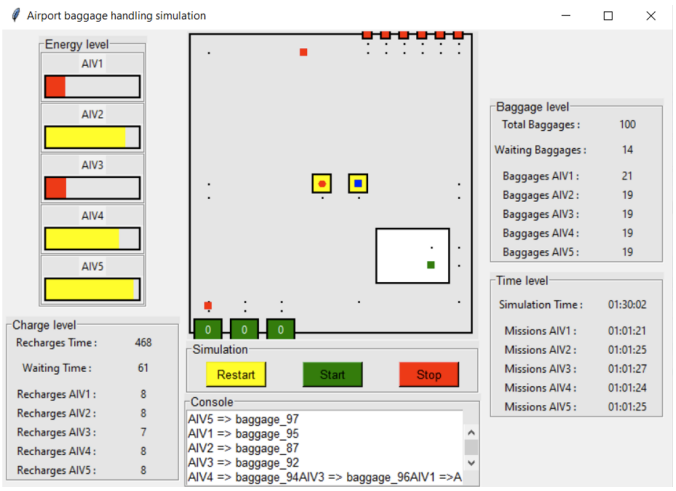


Figure 1. HCI of the simulation application

TA performed by a supervisor who questions AIVs to know their task completion costs.

The analysis is driven by three objectives aimed at optimizing TA: minimize x, maximize y, and minimize z. Where:

- x is the number of AIVs,
- y is the baggage throughput per hour,
- z is the recharge time of an AIV (in ideal conditions

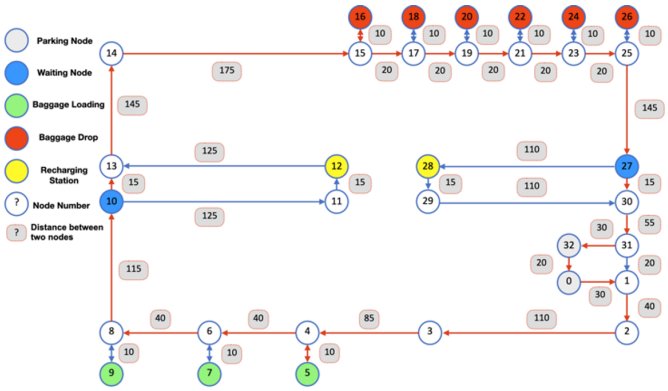


Figure 2. Oriented graph: distance in the environment in meters

where an AIV picks up one baggage per turn).

$$\begin{cases} 0 \leq x \leq \text{Max}(x) = \dfrac{L_{\text{avg}}}{d} \\ 0 \leq y \leq T_{\text{avg}} \times \text{Max}(x) \\ 0 \leq z \leq 3600 \\ z = \left(\dfrac{v_{\text{avg}} \times 3600}{C_{\text{bat}}} \right) \times (t_0 + t_1) \\ T_{\text{avg}} = \dfrac{v_{\text{avg}} \times (3600 - z)}{L_{\text{avg}}} \\ y = T_{\text{avg}} \times x \end{cases}$$

with:

- $Max(x)$ is the maximum number of AIVs.
- L_{avg} is the average length of the circuit.
- d is the safety distance between two AIVs.
- C_{bat} is the average capacity of a battery.
- t_0 is the average charging time of a battery.
- t_1 is the average waiting time for a battery recharge.
- T_{avg} is the average number of revolutions made by an AIV during one hour.
- v_{avg} is the average speed of AIVs on the circuit.

Through 8 scenarios, we will progressively introduce fuzzy inferences to determine the costs of task completion, battery recharging and speed adjustment.

A. The Simulation Framework

Fig. 3 presents the agent model proposed to test our dynamic task allocation strategies for AIVs in simulation. The objective is to have an agent-based modeling and simulation system designed generically to test different scenarios, but also different types of circulation plans. An infrastructure is deployed in the environment. It is composed of a circulation plan and active elements, such as beacons, tags, the two charging stations and the two types of treadmill for baggage entry and exit. Static or dynamic obstacles (e.g., operators) may be present in the environment. AIV fuzzy agents perform missions defined by paths on the traffic plan. AIV fuzzy agents are equipped with a radar to avoid collisions and have knowledge about the environment and other agents to operate and cooperate. AIV fuzzy agents communicate with each other with different types of standardized messages. AIV fuzzy agents have fuzzy and uncertain knowledge, but also incomplete and fragmented, in order to adapt to situations that are themselves uncertain. Baggage are objects managed by the environment: arrival flow on the entry treadmill, tracking of its localization, and exit from the circuit on the exit treadmill.

B. V2X Cooperation in the Simulations

To successfully complete their assigned tasks, AIVs must coordinate, cooperate, and share information about their tasks and environmental perceptions. They rely on ETSI messages, as described in Section II.D, to communicate their localization using CAM and report perceived obstacles using CPM and DENM, helping to prevent unexpected events. An additional type of V2V communication could further enhance cooperation among AIVs in task execution. For instance, if an AIV becomes blocked by an obstacle, breaks down, or is otherwise unable to complete its assigned task, it automatically sends a DENM. However, it would be beneficial for the AIV to also send a cooperative message, enabling it to delegate its task by providing the necessary information. In [42], we propose two new Cooperative Task Messages (CTM) designed specifically for task delegation. Similarly, [51] introduces a protocol with four new message types, including the Cooperative Response Message (CRM), which is used to communicate responses to cooperation requests. In our simulation model, AIV agents will use CRM messages as feedback to CTM messages, confirming their willingness to take on a delegated task. During the simulations, the sequence of communications between the supervisor, the AIV auctioneer, and multiple AIVs during the task allocation and reallocation process is illustrated in Fig.4. The supervisor initiates the process by sending a Cooperative Task Message (CTM) containing clustered tasks to an AIV acting as an auctioneer. The auctioneer then distributes these tasks by further clustering and auctioning them to other AIVs. This is done by sending a CTM [clustered auctioned tasks] to potential AIVs capable of performing the assigned duties. Once the auctioning phase is complete, the auctioneer allocates specific tasks by transmitting a CTM [allocated tasks] to the chosen AIVs. Each receiving AIV acknowledges the task assignment by sending a Cooperative Response Message (CRM) back to the auctioneer, confirming acceptance. Additionally, the same allocation mechanism is utilized for task reallocation. If an AIV encounters an obstacle, breaks down, or is unable to complete its assigned task, it can initiate a re-auction. In this scenario, the AIV itself takes on the role of an auctioneer, redistributing its pending tasks through CTM messages. The AIV that submits the most suitable bid will then integrate the reallocated tasks into its workload.

This structured communication process, visualized in Fig. 4, ensures effective task management and dynamic reallocation, enabling seamless cooperation among AIVs in an automated environment.

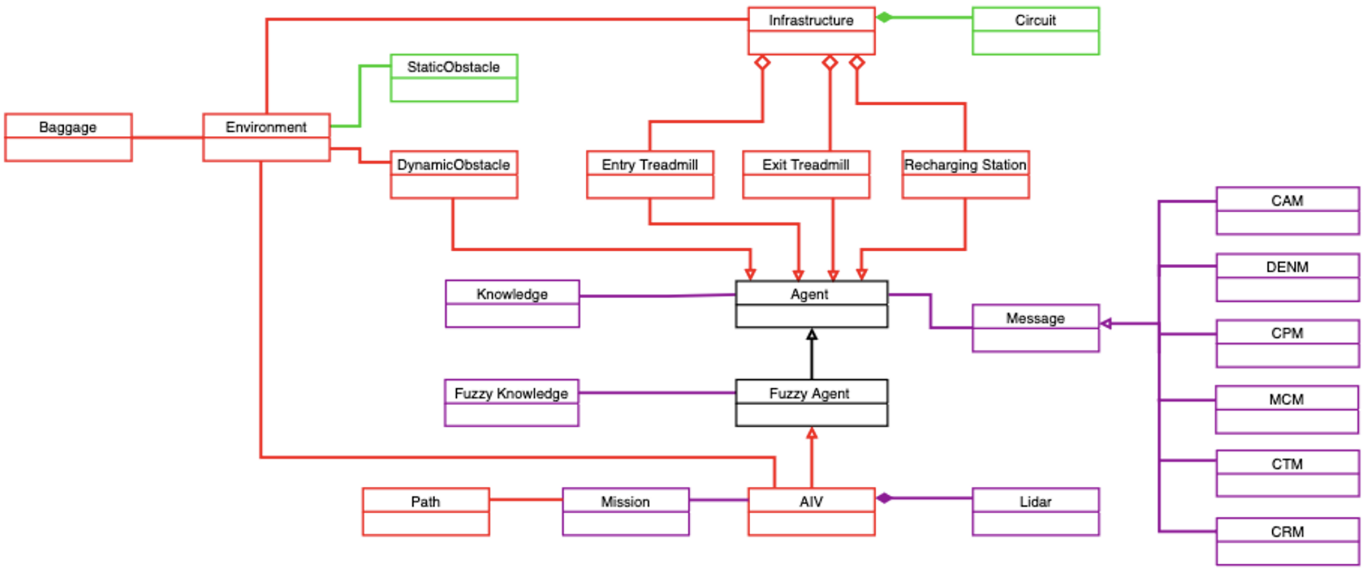


Figure 3. Simulator architecture: dynamic elements in red, static in green, and not related to the environment in purple.

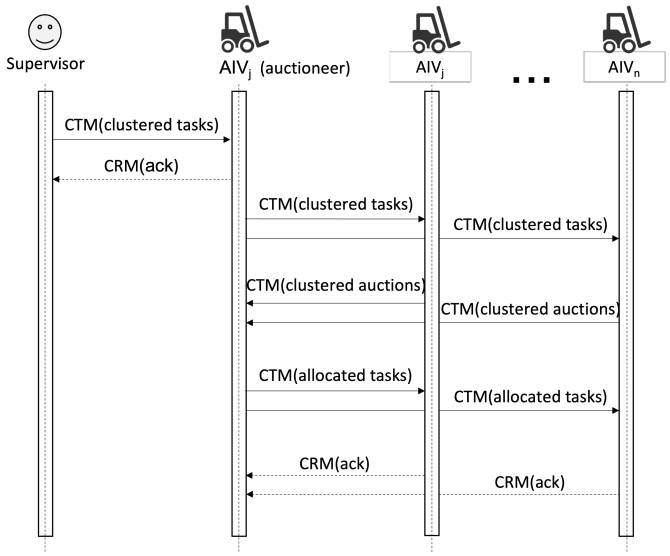


Figure 4. CTM and CRM exchanged during task allocation

C. Task Allocation with Basic Strategies

In this section, we provide a comparative analysis of three basic types of auction-based task allocation strategies: random TA, FIFO TA, and AIV availability-based TA. Each of these strategies is tested in a scenario:

- Sc1 (Random) is a TA scenario where missions are assigned to the AIV agents only randomly.
- Sc2 (FIFO) is a TA scenario where missions are assigned to AIV agents using a queuing mechanism.
- Sc3 (Available) is a TA scenario where missions are assigned to the most available AIV agents.

We simulated these three scenarios for 100 bags. We seek to minimize the maximum number of pending bags at a given time, the total simulation time, the average time to complete a mission per AIV agent, the number of missions completed per AIV agent during the simulation, and the activity rate per AIV agent. The simulation results are presented in Table II.

Random strategy: the maximum number of pending bags (19) is high, the simulation time is also high, and the allocation of missions and the activity rates of AIV agents are poorly balanced (the average activity rate at 0.72 is low). The random strategy does not allow allocation to AIV agents that are a priori available, which very quickly leads to pending bags being processed and therefore poor results.

FIFO strategy: this strategy brings a clear improvement in the results. The maximum number of pending bags (4) is very low, the simulation time is very correct, the allocation is almost uniform (only the stops for recharging the batteries cause imbalances), and the occupancy rate of the AIV agents is much better (0.84).

Available strategy: this strategy produces the best results, except for the maximum number of pending bags (8). Allocating a mission to an AIV agent that is more available than the others, improves the results. However, it is necessary to better manage the allocation based on pending bags and energy consumption to consolidate (or even optimize) this strategy.

D. Task Allocation with Fuzzy Strategies

In this section, we propose an analysis of task allocation by auction based on a fuzzy inference approach. As a reminder, fuzzy logic allows us to better understand natural, uncertain, imprecise and difficult to model phenomena by relying on the definition of if-then fuzzy rules and membership functions (linguistic variables) to fuzzy sets [52]. Two scenarios are studied. The first, Sc4, implements a TA strategy in which each

TABLE II
TASK ALLOCATION SIMULATION RESULTS IN SCENARIOS SC1, SC2, AND SC3, FOR 100 BAGS

Scenarios	Random	FIFO	Available
Maximum number of pending bags	19	4	8
Simulation time (s)	2270	1942	1846
Average mission time per AIV (in s)	[81, 81, 83, 83, 81]	[80, 82, 83, 81, 83]	[81, 80, 81, 83, 81]
Number of mission completed by AIV	[26, 26, 14, 14, 20]	[21, 21, 19, 21, 18]	[22, 21, 20, 19, 18]
Work rate per AIV	[0.93, 0.93, 0.51, 0.51, 0.71]	[0.87, 0.89, 0.81, 0.88, 0.77]	[0.97, 0.91, 0.88, 0.85, 0.79]

TABLE III
TASK ALLOCATION SIMULATION RESULTS IN SCENARIOS SC4 AND SC5, FOR 100 BAGS

Scenarios	Sc4	Sc5
Maximum number of pending bags	6	6
Simulation time (s)	1843	2000
Average mission time per AIV (s)	[80, 81, 80, 81, 82]	[81, 80, 81, 84, 83]
Number of missions completed by AIV	[21, 21, 21, 19, 18]	[23, 19, 21, 19, 18]
Work rate per AIV	[0.91, 0.92, 0.91, 0.84, 0.80]	[0.93, 0.76, 0.85, 0.80, 0.75]

TABLE IV
RECHARGE SIMULATIONS RESULTS IN SCENARIOS SC4 AND SC5, FOR 100 BAGS

Scenarios	Sc4	Sc5
Recharge time (s)	546	490
Waiting time for recharges (s)	34	16
Number of recharges	39	33
Distribution of recharges per AIV	[8, 8, 8, 8, 7]	[8, 6, 7, 6, 6]

AIV agent uses a fuzzy model with 3 linguistic input variables (availability of the AIV agent, distance from the baggage drop-off location, energy level of the AIV agent) to determine the cost of handling a mission (picking up and dropping off a baggage). The second, Sc5, takes the strategy of Sc4 and adds energy management with a second fuzzy model. With this new fuzzy model, the AIV agents determine whether they will need to recharge during a mission, which allows them to refine the calculation of the mission cost. The linguistic variables used in this scenario are: availability of the AIV agent, distance from the baggage drop-off location, energy level of the AIV agent, and distances of the 2 charging stations.

Fuzzy strategy in Sc4. The results presented in Table III and Table IV, with this new strategy are generally good: low maximum number of pending bags (6), good overall simulation time, good distribution of missions between AIV agents and good average AIV activity rate (0.88). However, a few elements of uncertainty are considered (3 linguistic variables at the input and one at the output). The introduction of other fuzzy elements (nuances in the simulation parameters) should improve the results, particularly in terms of maximum number of pending bags and management of battery recharges.

Fuzzy strategies in Sc5. In this new scenario, the raw results of the TA are slightly worse, as shown in Table III and Table IV, than in Sc4: same maximum number of pending bags (6), slightly longer overall simulation time, worse distribution of missions between AIV agents and worse average AIV occupancy rate (0.82). However, the overall recharge time is lower in this scenario, which can allow a greater availability of AIV agents (an area of improvement for the next scenarios).

E. Integration of Collision Avoidance and Speed Adaptation

Managing traffic situations on the circuit sometimes requires speed adaptations for AIVs. This is particularly the case for managing intersections between AIVs or for managing distances between AIVs. Obstacle avoidance may also need to be managed. However, although we have considered it in previous studies [41], we do not address it in this study. Fuzzy AIVs agents have fuzzy knowledge to adapt their speeds. This knowledge is mainly activated to respect the priority to the right when exiting baggage claim areas, when exiting baggage drop-off areas and when exiting battery charging areas. For this, four fuzzy linguistic variables were defined, three for the inputs to the fuzzy inference system (7, 8, 9) and one for the output (10):

- $AIV\_right\_distance$ (near, medium, far) (7)
- $AIV\_distance$ (near, medium, far) (8)
- AI_speed (slow, medium, fast) (9)
- $AIV\_speed\_adaptation$ (slow, medium, fast) (10)

The AIVs fuzzy inference system for adapting their speeds works through the activation of 15 fuzzy rules such as the following (11):
$IF\ AIV\_right\_distance\ IS$ near $AND\ AIV\_distance\ IS$ far $AND\ AIV\_speed\ IS$ medium $THEN\ AIV\_speed\_adaptation\ IS$ slow (11)

IV. IMPROVEMENT USING FUZZY HEURISTICS

Now, we propose to increase the relevance of previous auction TA scenarios based on a fuzzy inference approach, by integrating other types of realistic constraints concerning battery recharging and AIV agent speed adjustment made possible by a stronger knowledge of the fleet traffic and mission management context (increased awareness). Three scenarios are studied (Sc6, Sc7 and Sc8) to show that specific heuristics allow us to treat certain situations quite finely and to increase the collective/global performances of the AIV agents. The results are presented in Table V for task allocation and Table VI for battery recharging. Sc6 consists of completing scenario Sc5 to determine in which station the AIV agents can

recharge in order to minimize the waiting times for recharging, based on knowledge of the context of occupation of the stations and the needs of the other AIV agents (therefore more awareness for the agents). The linguistic variables used in this sixth scenario are the following: the availability of the AIV agent, the distance from the baggage drop-off location, the energy level of the AIV agent, the distances of the 2 recharging stations and the availability of the recharging stations. Sc7 takes up the strategy of Sc6 and adapts the recharging rate (80 or 100%) in order to increase their availability if the flow of incoming baggage increases and therefore if the number of pending bags is likely to increase. The linguistic variables used in this seventh scenario are: the availability of the AIV agent, the distance from the baggage drop-off location, the energy level of the AIV agent, the distances from the 2 charging stations, the availability of the charging stations and a variable energy charge rate (80 or 100%). Sc8 consists of increasing Sc7 by adapting/regulating the speed of the AIV agents according to the flow of baggage arrivals and therefore the potential increase in the number of pending bags to be processed, but also according to the speed, the proximity of other AIV agents (use of observed and safety distances). The linguistic variables used in this eighth scenario are as follows: the availability of the AIV agent, the distance from the baggage drop-off location, the energy level of the AIV agent, the distances of the 2 charging stations, the availability of the charging stations, a variable charging rate (80 or 100%) and urgency in relation to the number of pending bags.

Results of fuzzy inferences in Sc6. This is the implementation of a first heuristic to improve the TA but also the recharge decision. The objective is to minimize the waiting time for a recharge when an AIV agent must be available to take baggage. The results for TA are slightly better than in Sc5: the same maximum number of pending bags, a slightly shorter overall simulation time, a rather homogeneous average mission completion time, a better distribution of missions between AIV agents, and an average AIV activity rate that is roughly the same (0.82). However, if the overall recharge time is the same, the waiting time for recharges is significantly lower (14s).

Results of fuzzy inferences in Sc7. The second heuristic proposed in order to increase the availability of AIV agents so that they can take baggage according to their arrival flow while minimizing the waiting time for their recharges. In this scenario, the results for TA are significantly better than in the Sc6 scenario: the same maximum number of pending bags, but a shorter overall simulation time, a more homogeneous average mission completion time, a better distribution of missions between AIV agents and a higher average AIV activity rate (0.84). Regarding battery recharges, the results are of the same order for both scenarios: an identical overall recharge time, with in Sc7, a slightly higher waiting time for recharges (18s).

Results of fuzzy inferences in Sc8. A third heuristic was proposed in order to adjust speed of the AIV agents to minimize the maximum number of pending bags when the flow of baggage arrivals increases. The results for TA are much better than in Sc7: the same maximum number of pending bags, but a much lower overall simulation time (a consequence of the adaptation of speeds of AIV agents when necessary), an average time of completion of the missions and a distribution of the missions between the AIV agents always homogeneous, and finally, a lower average occupancy rate of the AIV agents (0.79), because the last two AIV agents are less requested due to the adaptation of the speeds of the first 3, in particular their increase in speed to respond to the increase in the flow of baggage arrivals. As for the battery recharges, the results are less good: the increase in the speeds of the AIV agents has an energy cost!

TABLE V
TASK ALLOCATION SIMULATION RESULTS IN SCENARIOS SC6; SC7 AND SC8 FOR 100 BAGS

Scenarios	Sc6	Sc7	Sc8
Maximum number of pending bags	6	6	6
Simulation time (s)	1964	1896	1675
Average mission time per AIV (s)	[79, 79, 80, 80, 81]	[79, 80, 80, 80, 80]	[67, 65, 67, 65, 67]
Number of missions completed by AIV	[22, 22, 20, 16, 20]	[22, 22, 21, 18, 17]	[22, 22, 22, 19, 15]
Work rate per AIV	[0.88, 0.88, 0.81, 0.65, 0.82]	[0.92, 0.93, 0.89, 0.76, 0.72]	[0.88, 0.85, 0.88, 0.74, 0.6]

TABLE VI
COMPARISON OF SCENARIOS SC6, SC7, AND SC8

Scenarios	Sc6	Sc7	Sc8
Recharge time	490	490	736
Wait time for recharges	14	18	119
Number of recharges	33	33	49
Distribution of recharges per AIV	[7, 7, 7, 5, 7]	[7, 7, 7, 6, 6]	[11, 11, 11, 9, 7]

V. CONCLUSION

We developed a multi-agent simulation platform to test different scenarios of task allocation management for mobile baggage conveyor robots (AIVs) in the context of Airport 4.0. This approach offers a flexible adaptation to the different aspects of AIV autonomy management and facilitates possible adjustments needed for deployment at an airport site. The use of a distributed multi-agent system provides temporary autonomy in case of central infrastructure failure, and can improve the management of individual AIV functions, such as task allocation, battery charging, collision avoidance, speed regulation, etc. To establish a basis for comparison of auction-based task allocation strategies with the fuzzy approach we wanted to develop, we started by defining three basic scenarios implementing random, FIFO and AIV availability strategies. We then tested a task allocation scenario with a basic fuzzy model incorporating cooperative V2X communication

and collision avoidance mechanisms. Then, we made several improvements to this scenario by extending the AIV's fuzzy decision model to: (1) recharging the AIVs batteries, (2) determining the recharging station, (3) determining the most relevant recharging rate, and (4) regulating the speed of the AIVs so that they adapt to the variation of the baggage arrival flow. The simulation results show that integrating adaptive fuzzy multi-agent models for managing task allocation, energy recharging management, determining the most favorable infrastructure elements (charging stations), cooperative task allocation through V2X, and speed adaptation with collision avoidance, can improve the operational efficiency of AIV fleet. The adaptive model not only improves task allocation and energy management but also ensures safer and more coordinated operations by dynamically adjusting speed and preventing collisions. These results highlight the importance of flexible and collaborative approaches to improve the performance of autonomous systems in dynamic environments. We plan to continue integrating fuzzy models into AIV agent behavior simulations and to add learning capabilities (e.g., based on neural networks [53]) to them in order to increase the relevance and efficiency of their decisions in the collective management of their autonomies. Moreover, to ensure our simulations better reflect real-world operations, we also plan to study the impact of unexpected events, such as security attacks or mis-behavior of AIVs, and to integrate corresponding mechanisms into our scenarios.

Acknowledgment

The authors would like to thank the Brittany region for funding the ALPHA project, as part of the PME 2022 call for projects entitled "Accelerate time to market of digital technological innovations from SMEs in the Greater West".

References

[1] Juliette Grosset et al. "Fuzzy Agent-Based Simulation of Integrated Solutions for Task Allocation and Battery Charge Management for Fleets of Autonomous Industrial Vehicles". In: *AI-based Systems and Services (AISyS) 2024*. Venice, Italy, Oct. 2024.

[2] Xiaolin Hu and Bernard P. Zeigler. "A Simulation-Based Virtual Environment to Study Cooperative Robotic Systems". In: *Integrated Computer-Aided Engineering* 12.4 (Jan. 2005), pp. 353–367. ISSN: 1069-2509. DOI: 10.3233/ICA-2005-12404.

[3] Naoum Tsolakis, Dimitrios Bechtsis, and Jagjit Singh Srai. "Intelligent Autonomous Vehicles in Digital Supply Chains: From Conceptualisation, to Simulation Modelling, to Real-World Operations". In: *Business Process Management Journal* 25.3 (Jan. 2018), pp. 414–437. ISSN: 1463-7154. DOI: 10.1108/BPMJ-11-2017-0330.

[4] Abdelfetah Hentout et al. "Human–Robot Interaction in Industrial Collaborative Robotics: A Literature Review of the Decade 2008–2017". In: *Advanced Robotics* 33.15-16 (Aug. 2019), pp. 764–799. ISSN: 0169-1864. DOI: 10.1080/01691864.2019.1636714.

[5] Peng Jing et al. "Agent-Based Simulation of Autonomous Vehicles: A Systematic Literature Review". In: *IEEE Access* 8 (2020), pp. 79089–79103. ISSN: 2169-3536. DOI: 10 . 1109 / ACCESS . 2020 . 2990295.

[6] Alain-Jérôme Fougères. "A Modelling Approach Based on Fuzzy Agents". In: *arXiv:1302.6442 [cs]* (Feb. 2013). arXiv: 1302.6442 [cs].

[7] Alain-Jérôme Fougères and Egon Ostrosi. "Fuzzy Agent-Based Approach for Consensual Design Synthesis in Product Configuration". In: *Integrated Computer-Aided Engineering* 20.3 (Jan. 2013), pp. 259–274. ISSN: 1069-2509. DOI: 10 . 3233 / ICA - 130434.

[8] Tuncer I. Ören and Nasser Ghasem-Aghaee. "Personality Representation Processable in Fuzzy Logic for Human Behavior Simulation". In: vol. Proceedings of SCSC 2003. Phoenix, AZ, USA, 2003-08-01/2003-07-31, pp. 3–10.

[9] Egon Ostrosi et al. "A Fuzzy Configuration Multi-Agent Approach for Product Family Modelling in Conceptual Design". In: *Journal of Intelligent Manufacturing* 23.6 (Dec. 2012), pp. 2565–2586. ISSN: 1572-8145. DOI: 10.1007/s10845-011-0541-5.

[10] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. "Multi-Robot Task Allocation: A Review of the State-of-the-Art". In: *Studies in Computational Intelligence*. Vol. 604. May 2015, pp. 31–51. ISBN: 978-3-319-18299-5. DOI: 10.1007/978-3-319-18299-5_2.

[11] Karthik Karur et al. "A Survey of Path Planning Algorithms for Mobile Robots". In: *Vehicles* 3.3 (Sept. 2021), pp. 448–468. ISSN: 2624-8921. DOI: 10.3390/vehicles3030027.

[12] Salma El-Ansary, Omar Shehata, and Elsayed Imam Morgan. *Airport Management Controller: A Multi-Robot Task-Allocation Approach*. Jan. 2017, p. 30. DOI: 10.1145/3029610.3029623.

[13] Matthias De Ryck, Mark Versteyhe, and Frederik Debrouwere. "Automated Guided Vehicle Systems, State-of-the-Art Control Algorithms and Techniques". In: (2020). DOI: 10.1016/j.jmsy.2019.12.002.

[14] Ahmed Hussein and Alaa Khamis. *Market-Based Approach to Multi-robot Task Allocation*. Dec. 2013. DOI: 10.1109/ICBR.2013.6729278.

[15] Stefano Mariani, Giacomo Cabri, and Franco Zambonelli. "Coordination of Autonomous Vehicles: Taxonomy and Survey". In: *ACM Computing Surveys* 54.1 (Apr. 2021), pp. 1–33. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3431231. arXiv: 2001.02443.

[16] William Ferreira, Fabiano Armellini, and Luis Antonio Santa-Eulalia. "Simulation in Industry 4.0: A State-

of-the-Art Review". In: *Computers & Industrial Engineering* (Sept. 2020), p. 106868. DOI: 10.1016/j.cie.2020.106868.

[17] Alaa Daoud et al. "ORNInA: A Decentralized, Auction-Based Multi-Agent Coordination in ODT Systems: Online Rescheduling with Neighborhood Exchange, Based on Insertion Heuristic and Auctions". In: *AI Communications* 34 (Jan. 2021), pp. 1–17. DOI: 10.3233/AIC-201579.

[18] Voemir Kunchev et al. *Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review*. Oct. 2006, p. 544. ISBN: 978-3-540-46537-9. DOI: 10.1007/11893004_70.

[19] Xiaoyan Yu and Marin Marinov. "A Study on Recent Developments and Issues with Obstacle Detection Systems for Automated Vehicles". In: *Sustainability* 12.8 (Jan. 2020), p. 3281. ISSN: 2071-1050. DOI: 10.3390/su12083281.

[20] Dae Hwan Kim, Nguyen Trong Hai, and Woong Yeol Joe. "A Guide to Selecting Path Planning Algorithm for Automated Guided Vehicle (AGV)". In: *AETA 2017 - Recent Advances in Electrical Engineering and Related Sciences: Theory and Application*. Ed. by Vo Hoang Duy et al. Vol. 465. Cham: Springer International Publishing, 2018, pp. 587–596. ISBN: 978-3-319-69813-7 978-3-319-69814-4. DOI: 10.1007/978-3-319-69814-4_56.

[21] Hang Ma and Sven Koenig. "Optimal Target Assignment and Path Finding for Teams of Agents". In: *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. AAMAS '16. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2016, pp. 1144–1152. ISBN: 978-1-4503-4239-1.

[22] Qazi Shaheen Kabir and Yoshinori Suzuki. "Comparative Analysis of Different Routing Heuristics for the Battery Management of Automated Guided Vehicles". In: *International Journal of Production Research* 57.2 (Jan. 2019), pp. 624–641. ISSN: 0020-7543. DOI: 10.1080/00207543.2018.1475761.

[23] Heiner Lasi et al. "Industry 4.0". In: *Business & Information Systems Engineering* 6.4 (Aug. 2014), pp. 239–242. ISSN: 1867-0202. DOI: 10.1007/s12599-014-0334-4.

[24] Matthias De Ryck, Mark Versteyhe, and Keivan Shariatmadar. "Resource Management in Decentralized Industrial Automated Guided Vehicle Systems". In: *Journal of Manufacturing Systems* 54 (Jan. 2020), pp. 204–214. DOI: 10.1016/j.jmsy.2019.11.003.

[25] Mauricio F. Jaramillo Morales and Juan B. Gómez Mendoza. "Mixed Energy Model for a Differential Guide Mobile Robot". In: *2018 23rd International Conference on Methods & Models in Automation & Robotics (MMAR)*. Miedzyzdroje, Poland, 27/2018-08-30, pp. 114–119. DOI: 10.1109/MMAR.2018.8486117.

[26] Xiaolong Zhang et al. "Optimal Trajectory Planning for Wheeled Mobile Robots under Localization Uncertainty and Energy Efficiency Constraints". In: *Sensors* 21.2 (Jan. 2021), p. 335. ISSN: 1424-8220. DOI: 10.3390/s21020335.

[27] Alexandr Štefek et al. "Optimization of Fuzzy Logic Controller Used for a Differential Drive Wheeled Mobile Robot". In: *Applied Sciences* 11.13 (Jan. 2021), p. 6023. ISSN: 2076-3417. DOI: 10.3390/app11136023.

[28] Gürkan Gürgöze and İbrahim Türkoğlu. "A Novel, Energy-Efficient Smart Speed Adaptation Based on the Gini Coefficient in Autonomous Mobile Robots". In: *Electronics* 11.19 (Jan. 2022), p. 2982. ISSN: 2079-9292. DOI: 10.3390/electronics11192982.

[29] Zhaoxiang Tan et al. "Adaptive Partial Train Speed Trajectory Optimization". In: *Energies* 11.12 (Dec. 2018), p. 3302. ISSN: 1996-1073. DOI: 10.3390/en11123302.

[30] Chengyuan Ma et al. "Signal Timing at an Isolated Intersection under Mixed Traffic Environment with Self-Organizing Connected and Automated Vehicles". In: *Computer-Aided Civil and Infrastructure Engineering* 38.14 (2023), pp. 1955–1972. ISSN: 1467-8667. DOI: 10.1111/mice.12961.

[31] Qianwen Li et al. "Simulation of Mixed Traffic with Cooperative Lane Changes". In: *Computer-Aided Civil and Infrastructure Engineering* 37.15 (2022), pp. 1978–1996. ISSN: 1467-8667. DOI: 10.1111/mice.12732.

[32] Zhao Zhang et al. "Traffic Signal Optimization for Partially Observable Traffic System and Low Penetration Rate of Connected Vehicles". In: *Computer-Aided Civil and Infrastructure Engineering* 37.15 (2022), pp. 2070–2092. ISSN: 1467-8667. DOI: 10.1111/mice.12897.

[33] Albashir A. Youssef et al. "Brief Survey on Industry 4.0 Warehouse Management Systems". In: *International Review on Modelling and Simulations (IREMOS)* 15.5 (Oct. 2022), pp. 340–350. ISSN: 2533-1701. DOI: 10.15866/iremos.v15i5.22923.

[34] Alicja Dabrowska, Robert Giel, and Sylwia Werbinska-Wojciechowska. "Human Safety in Autonomous Transport Systems – Review and Case Study". In: (2021), pp. 57–71. ISSN: 18958281. DOI: 10.2478/jok-2021-0005.

[35] Andras Kokuti et al. "V2X Communications Architecture for Off-Road Autonomous Vehicles". In: *2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. June 2017, pp. 69–74. DOI: 10.1109/ICVES.2017.7991903.

[36] Juliette Grosset et al. "Collective Obstacle Avoidance Strategy - an Agent-Based Simulation Approach". In: *ASPAI 2022: 4th International Conference on Advances in Signal Processing and Artificial Intelligence*. Corfu, Greece, Oct. 2022.

[37] ITS. *Intelligent Transport Systems; Vehicular Communications; Basic Set of Applications; Part*

3: Specification of Decentralized Environmental Notification Basic Service, ETSI EN 302 637-3 V1.2.1. Tech. rep. 2014.

[38] ITS. *Intelligent Transport Systems; Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS), Release 2, ETSI TR 103 562 V2.1.1*. Tech. rep. 2019.

[39] Haotian Shi et al. "A Distributed Deep Reinforcement Learning–Based Integrated Dynamic Bus Control System in a Connected Environment". In: *Computer-Aided Civil and Infrastructure Engineering* 37.15 (2022), pp. 2016–2032. ISSN: 1467-8667. DOI: 10.1111/mice.12803.

[40] Ngai Meng Kou et al. "Multi-Agent Path Planning with Non-constant Velocity Motion". In: (2019).

[41] J. Grosset et al. "A Cooperative Approach to Avoiding Obstacles and Collisions between Autonomous Industrial Vehicles in a Simulation Platform". In: *Integrated Computer-Aided Engineering* 30.1 (Jan. 2023), pp. 19–40. ISSN: 1069-2509. DOI: 10.3233/ICA-220694.

[42] J. Grosset et al. "Multi-Agent Simulation of Autonomous Industrial Vehicle Fleets: Towards Dynamic Task Allocation in V2X Cooperation Mode". In: *Integrated Computer-Aided Engineering* 31.3 (Apr. 2024), pp. 249–266. ISSN: 10692509, 18758835. DOI: 10.3233/ICA-240735.

[43] Vivek Yerubandi et al. "Navigation System for an Autonomous Robot Using Fuzzy Logic". In: *International Journal of Scientific and Research Publications (IJSRP)* 5 (Feb. 2015), pp. 1–4.

[44] Andry Meylani et al. *Different Types of Fuzzy Logic in Obstacles Avoidance of Mobile Robot*. IEEE. Piscataway, NJ, USA, Oct. 2018, p. 100. DOI: 10.1109/ICECOS.2018.8605206.

[45] Bhumeshwar Patle et al. "A Review: On Path Planning Strategies for Navigation of Mobile Robot". In: *Defence Technology* 15.4 (Aug. 2019), pp. 582–606. ISSN: 2214-9147. DOI: 10.1016/j.dt.2019.04.011.

[46] Amir Nasrinahar and Joon Huang Chuah. "Intelligent Motion Planning of a Mobile Robot with Dynamic Obstacle Avoidance". In: *Journal on Vehicle Routing Algorithms* 1 (Jan. 2018). DOI: 10.1007/s41604-018-0007-4.

[47] Marwan Alakhras, Mourad Oussalah, and Mousa Hussein. "A Survey of Fuzzy Logic in Wireless Localization". In: *EURASIP Journal on Wireless Communications and Networking* 2020.1 (May 2020), p. 89. ISSN: 1687-1499. DOI: 10.1186/s13638-020-01703-7.

[48] Min-Fan Ricky Lee and Asep Nugroho. "Intelligent Energy Management System for Mobile Robot". In: *Sustainability* 14.16 (Jan. 2022), p. 10056. ISSN: 2071-1050. DOI: 10.3390/su141610056.

[49] Juliette Grosset et al. "Fuzzy Multi-Agent Simulation for Collective Energy Management of Autonomous Industrial Vehicle Fleets". In: *Algorithms* 17.11 (Nov. 2024), p. 484. ISSN: 1999-4893. DOI: 10.3390/a17110484.

[50] Egon Ostrosi, Alain-Jérôme Fougères, and Michel Ferney. "Fuzzy Agents for Product Configuration in Collaborative and Distributed Design Process". In: *Applied Soft Computing* 12.8 (Aug. 2012), pp. 2091–2105. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2012.03.005.

[51] Michael R. Hafner et al. "Cooperative Collision Avoidance at Intersections: Algorithms and Experiments". In: *IEEE Transactions on Intelligent Transportation Systems* 14.3 (Sept. 2013), pp. 1162–1175. ISSN: 1558-0016. DOI: 10.1109/TITS.2013.2252901.

[52] L.A. Zadeh. "Fuzzy Sets". In: *Information and Control* 8.3 (June 1965), pp. 338–353. ISSN: 00199958. DOI: 10.1016/S0019-9958(65)90241-X.

[53] Hendra Yudha et al. *Performance Comparison of Fuzzy Logic and Neural Network Design for Mobile Robot Navigation*. IEEE. Piscataway, NJ, USA, Oct. 2019, p. 84. DOI: 10.1109/ICECOS47637.2019.8984577.

Implementation of a Predictive AI to Feed Simulation

Carlo Simon, Merlin Hladik, and Natan Georgievic Badurasvili
Hochschule Worms
Erenburgerstr. 19, 67549 Worms, Germany
e-mail: {simon,hladik,natan.badurasvili}@hs-worms.de

***Abstract*—This paper extends a former version presented at the SIMUL 2024 conference. Its topic resulted from a remark of an industry partner dissatisfied with the result of a traffic simulation of a warehouse. Although the simulation was a replica of the behavior on base of the current order data, it deviated from the real situation since many distributors do not adhere to their orders. Methods of predictive artificial intelligence and especially machine learning have been identified to adapt the simulation input on the base of past schedules. The paper answers the question on how to extend the previous simulation model by a suitable forecast component and explains the implementation in more detail than the original paper. Unfortunately, the industry partner does not collect the data needed for such a forecast. Therefore, test data was generated which is explained, too. By the example of a real-world warehouse scenario, the simulation of its traffic and the information needed for this is demonstrated. Afterwards, the necessary extensions of the data set are explained and how to set up a machine learning component to predict future deviations of schedules. The adapted schedules can then be simulated to create alternative schedules. Like in a construction manual, the implementation is explained step by step.**

***Keywords*-Predictive Artificial Intelligence; Neural Networks; Machine Learning; Simulation; Petri Nets.**

I. Introduction

The idea to use AI to feed a simulation [1] was initiated by discussions during the SIMUL 2023 conference [2]. The participants had various imaginations about the impact of currently discussed Artificial Intelligence (AI) methods like transformers on simulation. While simulation is about causality, many AI methods are about correlation. The participants generally doubted that AI will substitute current simulation methods. However, there is still the possibility to pair both approaches.

This discussion occurred at the right time, as there was a need to address a problem that had arisen in an industry project using alternative methods. They could simulate the incoming and outgoing traffic of a large warehouse for registered transports as described in Section II with the aid of a high-level Petri net. But this did not take into account that the registered (planned) arrival time and the actual arrival time of the transports often do not match.

An investigation began into possible reasons for late transports, such as transport distance, the shipping agent, the producer, or current weather conditions. None of these factors were considered during the actual simulation of the warehouse's inbound and outbound traffic.

Unfortunately, the industry partner does not collect information about delays and, hence, cannot provide empirical values. Nonetheless, there was a desire to further develop the simulation environment by incorporating AI technologies. This idea led to the research question:

> *How can we extend our simulation model by a forecast component based on machine learning?*

A demonstration of the feasibility of this approach has been presented at the SIMUL 2024 conference [1]. The extension within this paper explains the implementation in detail.

Before explaining the new AI specific components of the simulation environment, the paper continues in Section II with a description of the high-level Petri net simulation model for the inbound and outbound traffic of a warehouse. Particular attention is given to the data required for simulation. Afterwards, in Section III, a brief introduction is given to the necessary Machine Learning (ML) methods. Both topics are combined in Section IV to define the structure of a possible ML data set for the problem first and to explain the ML approach in Section V next. The implementation of the ML component is explained in three steps in Sections VI, VII, and VIII. Section IX demonstrates how to integrate the ML approach into the existing environment. The paper ends in Section X with a conclusion and an outlook on future work.

II. Real-World Problem

The industry partner at an industrial park (Industriepark Höchst, *ISL*) provides logistics services for chemical, pharmaceutical and healthcare industries and currently expands its logistics services. Freely published information show the size of this venture [3]:

- Space for more than 21,000 pallets
- 9 separate warehouse sections
- Storage of multiple hazardous material storage classes for chemicals and pharmaceutics
- A wide temperature range from -6 to 20 degrees Celsius in the different sections of the warehouse

As depicted in Figure 1, the warehouse can be accessed via ramps (2). Each section has a loading zone (3) and the actual storage zone (6).

During planning and go live, the inbound and outbound traffic of this warehouse has to be simulated to objectify assumptions made during the conceptual phase. A typical inbound process is conducted as follows: before approaching the industrial park, the shipping agent books a time slot in advance. When trucks arrive, the drivers register with

the gatekeeper in the office (1). Afterwards, they dock at the ramp they have been assigned to (2). Then, the goods are picked by standard forklifts (called VHS) and placed in the staging zone (3). After the truck has left, the goods are carried through the driving zone (4) and dropped on a handover point (5). Finally, narrow aisle forklifts (called SGS) pick the goods and store them in the high rack storage area (6).

Outbound processes are executed in reverse order, except that the goods are provided in the staging zone before trucks arrive.

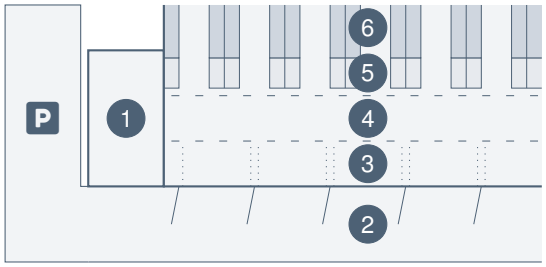


Figure 1. Sketch of the Warehouse.

Various specifics may be excluded from the simulation, such as the commissioning of certain goods. Furthermore, the precise positioning of goods within the warehouse, and consequently the exact driving times, hold diminished importance. In lieu of this, reasonable average times have been selected to replicate typical operational norms.

Model and simulation have been described in detail in [2] and [4]. The model consists of state machines for the inbound and outbound processes and a high-level Petri net to execute these state machines in parallel. Also, time constraints and restricting resources are taken into account.

Two conceptual data models for orders and resources are needed for this. The resources are discussed, first. Table I shows the data needed for simulation based on one kind of resource. This information is spread over several tables (or, in terms of Petri nets, over several places) for each kind of resource.

TABLE I. Attributes for the Resource Places

Attribute	Description
id	id for resources of this kind
product	product group
free	available or locked
timestamp	timestamp of the latest state change
order	assigned order id

Attribute *id* may identify a specific resource like a numbered ramp or a dedicated resource of this kind, like one of the VHS. *product* describes the resource allocation to chemical or pharmaceutical. *order* references to the order that uses this resource and simplifies joins among the different data sets.

Table II shows the data needed for orders. Beside an identifier *id* and the specification whether the order belongs to a chemical or pharmaceutical *product*, the *total* number of pallets for the transport is specified. *status* identifies the current order's state and, implicitly, whether this is an inbound or outbound process.

TABLE II. Attributes of an Order's State

Attribute	Description
id	order id
product	product group
total	total amount of pallets requested
status	initial or current order status
ramp	target ramp
arrival	scheduled time of arrival
preparation	scheduled time of completed staging
fillHandover	amount of pallets in handover areas
fillRamp	amount of pallets at target ramp
fillTruck	amount of pallets in truck
usedGate	used resource *gate*
usedSGS	used resource *SGS*
usedVHS	used resource *VHS*
timestamp	timestamp of the latest state change

One or two times must be defined per order: for both inbound and outbound, the *arrival* time of the truck is given due to its registration. Outbound processes additionally need a *preparation* time when staging begins. This staging time leaves room for optimisation for the warehouse operators.

At the end of the simulation explained in the following, all changes to orders are exported to a dashboard. A *timestamp* traces the moments these changes occur. To simplify the computing of this visualisation, the allocated resources like *ramp*, *usedGate*, *usedSGS*, *usedVHS* are saved. Finally, the amount of goods at the different places is stored in the attributes *fillHandover*, *fillRamp*, and *fillTruck*.

Without having an impact on the result, the real process and the simulated one differ slightly. For the simulation, the ramps are assigned in advance; in real world the gatekeeper decides on the ramp based on personal experience.

The *Process-Simulation.Center (P-S.C)* was chosen for modelling and simulation [5]. Since the *P-S.C* is a Petri net based Integrated Management System, it fulfils further constraints important for the industry partner among the pure ability to simulate. Safety and security aspects are of high priority to ISL. Therefore, access to and visibility of (real-world) data must be limited.

The *P-S.C-Cloud* (the *P-S.C* is only provided via internet) and the multi-client capability of the *P-S.C* help to manage security issues [5]: The tool itself only runs locally in a browser with data never leaving the system. Sensitive input data and simulation results can be stored on in-house servers without the *P-S.C-Cloud* ever coming in contact.

In addition, the industry partner requires a user interface (UI) to edit the simulation parameters (orders, times, resources, priorities) easily. The simulation results ought to be presented in a descriptive dashboard. Today, such an

interface is called a digital shadow, a piece of software which maps real-world data and processes to a virtual world [6].

Figure 2 explains the interaction between the *P-S.C* and the local UI using CSV-files. Simulation parameters can be imported this way, too.

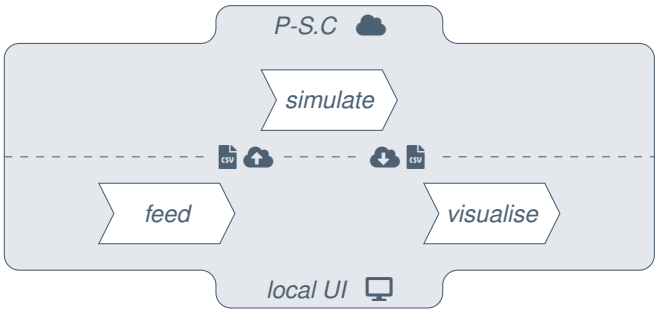


Figure 2. Coupling of Digital Shadow (local UI) and *P-S.C*.

The execution of a specific simulation scenario is divided into three distinct phases:

- **feed:** The local UI is implemented as a web-based application that can be used with any internet connected computer or iPad. It is used to enter the simulation parameter, priorities and especially the actual order data.
- **simulate:** The *P-S.C* loads the data into the browser of the end user to start the simulation. The *P-S.C-Cloud* does not get in touch with it which guarantees full control over the data. After the simulation has finished, an automatic download is started and the users can store the results on local hardware.
- **visualise:** The downloaded file in turn can now be uploaded in a dashboard which is also implemented in the local UI. This helps the end-users to find the best strategy for driving the warehouse. In particular, workload spikes, transportation bottlenecks, and staff occupancy can be analysed and visualised.

This approach is limited to simulate one specific schedule of orders for one specific period, e.g., one day. It is not flexible concerning deviations of the transports. The next sections explain how to extend this environment to handle this limitation.

III. Machine Learning Fundamentals

Artificial Intelligence (AI) is defined in many different ways. The most common definition of AI is that of a rational agent. Such an agent tries to provide the best (expected) outcome, given its inputs. What constitutes the best outcome remains a matter of definition [7].

If a rational agent is created to improve the provided output by gaining some sort of experience, this is called Machine Learning (ML). Figure 3 shows the idea of ML as described by [8]. The figure depicts the rational agent as a model. Its task (red) is to compute an output when presented with input data. The model obtains a mapping based on training data. Its formation is the learning problem (blue). Which algorithms are used depends on the chosen ML model [8].

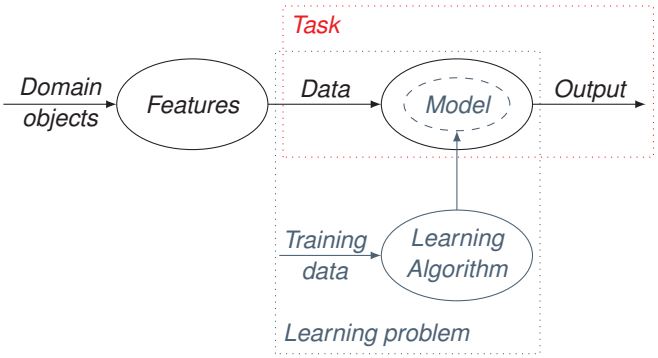


Figure 3. Machine Learning as explained by [8].

One possible ML model is Deep Learning (DL). A DL model is an artificial neural network with several (hidden) layers. A neural network is to mimic the principle of real neurons. Single neurons are connected by directed, weighted arcs. If the incoming arcs yield a high enough activation potential, the activation function in a neuron triggers a corresponding output. The weighs constitute the main parameters of this ML model type [9].

If the training set includes results, this is called supervised learning. In this case, the data is considered labelled. The metric used to examine the model's quality is to compare the actual results with the computed ones [10].

Two kinds of problems may be solved with the aid of artificial neural networks:

- **Classification**: In mathematical notation, a classifier is a function $y = f(x)$, where x is the input data item and y is the output category [11].
  Applied to our example of a warehouse, classification could help to predict which transports may be unpunctual.
- **Regression**: In regression, we try to understand the data points by discovering the curve that might have generated them [11].
  Applied to our example of a warehouse, regression could help to predict how many minutes transports may be unpunctual.

IV. Extended Data Set for ML

The aim of the presented approach is to correct and complete future orders and especially to forecast

- which future transports will or will not be in time, i.e., to solve a classification problem, and
- to forecast the expected deviation of the arrival time of future transports, i.e., to solve a regression problem.

But which parameters may influence the arrival time of trucks? And how can this information be coded, such that it can be processed by a learning algorithm?

First, it is worth to consider the data used for simulation already. All "internal" parameters of the orders like order *id*, current *timestamp* of the simulation, allocated *ramp* and other resources (*fillHandover*, *fillRamp*, *fillTruck*, *usedGate*, *usedSGS*, or *usedVHS*), and the *preparation* time for outgoing orders can be ruled out as possible sources.

The probably most valuable source is the planned *arrival* time coded as a timestamp consisting of date and time. This information, which is typically stored as a string, should be preprocessed for a learning algorithm. The following information can be extracted and coded as discrete numbers:

- The month or season of arrival, which may have an impact due to weather conditions.
- The arrival time in hours or at least in categories like early, in the middle of the day or by the end of the day, which may have an impact on the transportation conditions like traffic or the work schedule of the drivers.
- The day of week which may have an impact on the traffic intensity.

Also, the *state* attribute may be of interest because it is used to differ between incoming and outgoing transports. For outgoing transports, "only" an empty truck has to arrive while incoming transports need to be prepared before they reach the warehouse. This might have an impact on the arrival time.

Attribute *product* is not as valuable as supposed by its name. The simulation only distinguishes between chemical and pharmaceutical products which is very rough. It is not obvious that these two categories correlate with a deviation since there are hundreds of different concrete products that belong to these two groups.

Finally, the *total* amount of pallets is a candidate which might have an impact on the arrival time, especially for incoming transports. Large amounts of goods may cause more problems when being loaded compared to smaller ones.

Further attributes might have an impact which are not considered during simulation. The following attributes have been identified:

- The transportation *distance*, especially if a truck has to arrive from outside of the industrial park or whether it is an internal transport.
- The *shipping agents* may differ concerning their quality standards and the accuracy of delivery time.
- Finally, the producer might have an impact on the time a transport can start after being loaded and, hence, whether it arrives on time.

All remaining values can be enumerated and can hence be used for training of a neuronal net.

Unfortunately, the industry partner does not track these additional information. Even the deviation between the planned and the actual arrival time is not documented. Hence, the following considerations are only of theoretical nature.

V. ML for Predictive Scheduling

In the context discussed in this paper, Deep Neural Networks (DNNs) can be used threefold. First, they can create yet missing data, thus establishing a means to plan ahead of knowledge. Second, they can predict which transports may be not on time. Third, they can surmise time deviations.

A. Neural Nets to Complete the Feed

Data becomes worse - or even non-existent - the farther the look into the future. Thus, missing but plausible data has to be integrated into the feed. The corresponding DNN gets trained with historical data of planned arrival times of trucks. From this training data, the net creates fictional yet plausible data entries to complete the fragmentary known data. The thus enriched data constitutes one possible scenario to be analysed by simulation. As it is the first fully scheduled data set, it can be regarded as the base scenario.

B. Neural Nets for Classification

The associated DNN can learn from historical data which deliveries and dispatches were on time. Thus, it can predict the probability of a trip to be delayed or advanced. This is due to the neural nets capability to learn from underlying correlations. Such correlations may be interpreted as questions like:

- Are there shipping agents that often are late?
- Are there producers that always are early?
- Are midweek deliveries more reliable than ones on Mondays?

After establishing the probabilities of unpunctuality, alternative scenarios can be established. These scenarios incorporate differing yet still plausible delay and advance times. The corresponding schedules can then be compared to the base scenario.

C. Neural Nets for Regression

Knowing which delivery may be late is one side of the coin. The other is the actual time. The fitting DNN can make guesses about these times. Again, the capabilities of neural nets to represent correlations prove useful: Several different effects may overlap that may add up to massive delays. An example for this may be an unreliable hauler in the midst of winter on an early Monday morning. These time delta represent the last puzzle piece in creating alternative scenarios.

D. Implementation Insights

As a tool only provided via internet, the architecture of the *P-S.C-Cloud* consists of a JavaScript client and a PHP Server. The prototypical implementation of a ML component to conduct the previously described tasks is done with Python instead. The reason for this is the existence of a large number of ML libraries that can simply be combined. Especially the following packages are used [12]:

- **NumPy** provides data types and functions for easier handling of complex structures, such as vectors and matrices.
- **pandas** is designed for more complex structures and their easy handling. One strength is its extensive functionality for table structures.
- **Matplotlib** is used for visual analyses and plotting.
- **scikit-learn** contains many ML algorithms that can be easily used in your own program.
- **Keras** can build artificial neural networks.
- **TensorFlow** extends Keras with additional well performing functionalities and can handle large and complex data structures.

With the goal to train a DNN as an example, typical delivery information including possible delays, have been guessed in a students' project. While some of the information have been randomised, others include pattern like specific shipping agents always being late. The implementation for this is explained next in Section VI in more detail.

The numerical representation of the input data is partially generated in the JavaScript part of the implementation, others make use of the predefined ML algorithms in Python.

The derived information concerning completed order lists, classification and regression are currently produced in the Python environment. They can be stored in CSV files which can then be integrated into the *P-S.C-Cloud* via file upload. The corresponding implementations are explained in Section VII and Section VIII.

VI. Synthetic Data

The data generation is the first step of the implementation. Like the other two steps, the implementation is explained in great detail ensuring that anyone can rebuild the solution from scratch.

A. Description of Data

The following categories have been build:

- **total:** the total number of pallets in a delivery which may influence the transport time.
- **producer:** manufacturer of the order delivery which might affect the departure time of a transport after loading and hence the punctuality of the arrival.
- **carrier:** carrier for the delivery of the order, because they may differ in terms of quality standards and delivery accuracy.
- **weekday:** the traffic intensity may vary at the different days of the week.
- **season:** seasonal differences of the weather conditions may influence the traffic.
- **distance:** the distance of a delivery from the manufacturer to the destination.
- **arrival:** the respective delivery time.
- **delay:** delay times in four classes. The aim is to predict delays depending on the aforementioned parameters.

The next step is to generate this data as a CSV file using Python code.

B. Import of Required Modules

The **pandas** and **NumPy** modules are to be imported at the beginning of the code.

```
# Import of required modules
import pandas as pd
import numpy as np
```

In this context, **NumPy** provides the tools for generating random values to simulate various input features of the data set.

The **pandas** library is utilized to organize the generated data into a structured DateFrame format which resembles a table. This format is suitable for further processing, exporting to a CSV file, and use in machine learning workflows. **pandas** allows for easy labeling of columns and transformation of categorical and numerical data into a unified structure.

C. Consistent Data Generation

The size of the data set is determined next. Also, the random seed is set to 42 to ensure that the generated data remains consistent each time the program is invoked.

```
# Determination of the size
number_of_records = 1000
np.random.seed(42)
```

Afterwards the categories and possible values are defined.

```
_number_of_pallets = np.random.randint(1, 24, size =
      number_of_records)
_producer_classes = ['prod1','prod2','prod3']
_carrier_classes = ['car1','car2','car2']
_weekdays = np.random.randint(1, 7, size =
      number_of_records)
_seasonal_classes = [1, 2, 3, 4]
_seasons = []
_transport_distance_classes = [250,500,750]
_transport_distances = []
_arrival_time_classes = ['morning','midday','evening']
_arrival_times = []
_delay_time_classes = [0,15,30,45]
_delay_times = []
```

In the subsequent phase, the carriers are allocated according to their associated probabilities. Each carrier is generated with an identical probability, ensuring an even distribution.

```
_carriers = np.random.choice(_carrier_classes,
      number_of_records, p=[0.3333333333333333,
      0.3333333333333333, 0.3333333333333333])
```

D. Delay Times, Transport Distances, and Arrival Times

After the carriers are assigned, delay times are generated. The objective is to create shorter delays for `car1`, longer delays for `car2` and moderate delays for `car3`. By creating varying delay times for each carrier, patterns can be identified in the data, which are essential for the AI's predictions.

```
def generate_delay_time(_carriers):
  if _carriers == 'car1':
    return np.random.choice(_delay_time_classes, p=[0.6,
      0.2, 0.1, 0.1])
  elif _carriers == 'car2':
    return np.random.choice(_delay_time_classes, p=[0.1,
      0.1, 0.4, 0.4])
  elif _carriers == 'car3':
    return np.random.choice(_delay_time_classes, p=[0.1,
      0.4, 0.4, 0.1])

for i in _carriers:
  _delay_times.append(generate_delay_time(i))
```

Next, transport distances are assigned to the different carriers. `car1` records have a transport distance of 250 km, `car2` 500 km, and `car3` 750 km.

```
def generate_transport_distance(_carriers):
  if _carriers == 'car1':
    return _transport_distance_classes[0] # 250 km
  elif _carriers == 'car2':
    return _transport_distance_classes[1] # 500 km
  elif _carriers == 'car3':
    return _transport_distance_classes[2] # 750 km

for i in _carriers:
  _transport_distances.append(generate_transport_distance(i
      ))
```

The three carriers also vary concerning their arrival times with `car1` deliveries usually arrive in the morning, `car2` midday, and `car3` in the evening.

```
def generate_arrival_time(_carriers):
  if _carriers == 'car1':
    return _arrival_time_classes[0] # midday
  elif _carriers == 'car2':
    return _arrival_time_classes[1] # morning
  elif _carriers == 'car3':
    return _arrival_time_classes[2] # evening

for i in _carriers:
  _arrival_times.append(generate_arrival_time(i))
```

E. Generating Seasons

Season dependent delay times represent delays caused by weather conditions.

```
def generate_season(_delay_times):
  if _delay_times == 45:
    return _seasonal_classes[3] # Winter
  elif _delay_times == 30:
    return _seasonal_classes[2] # Autmn
  elif _delay_times == 15:
    return _seasonal_classes[0] # Spring
  elif _delay_times == 0:
    return _seasonal_classes[1] # Summer

for i in _delay_times:
  _seasons.append(generate_season(i))
```

F. Creating the Data Structure

To complete the data generation, the data structure is created as a DataFrame and afterwards exported as a CSV.

```
data = pd.DataFrame({
  'total' : _number_of_pallets,
  'firm' : np.random.choice(_producer_classes,
     number_of_records),
  'carrier' : _carriers,
  'weekday' : _weekdays,
  'season' : _seasons,
  'distance' : _transport_distances,
  'arrival' : _arrival_times,
  'delay' : _delay_times
})

data.to_csv('delivery_data_auto.csv', index=False)
```

VII. Classification

The task is to use the previously generated data as training and test data in order to classify a possible delay in future deliveries.

A. Import of Required Modules

To implement the neural network and prepare the data set for training, several essential Python libraries and modules are utilized:

```
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import numpy as np
```

Section V includes a short description of these libraries.

B. Import CSV Files

To use the previously generated data, the CSV file is imported as a DataFrame.

```
path1 = "delivery_data_auto.csv"
data = pd.read_csv(path1, delimiter=',')

# Output of the first five rows
print(data.head())
```

Figure 4 shows the first five rows of the data set.

Id	total	firm	carrier	weekday	season	distance	arrival	delay
0	7	prod1	car2	4	4	500	midday	45
1	20	prod1	car3	3	4	750	evening	45
2	15	prod3	car1	3	2	250	morning	0
3	11	prod2	car2	1	3	500	midday	30
4	8	prod2	car1	6	2	250	morning	0

Figure 4. Output of the first five rows.

Also, the data to which the AI makes a prediction is imported as a DataFrame as exemplarily shown in Figure 5. The record describes an order of 16 pallets of producer `prod3` carried out by the freight carrier `prod2` in winter, arriving midday on a Wednesday with a delivery-distance of 500 km.

```
Id total firm  carrier weekday season distance arrival
0     16 prod3    car2       3      4      500  midday
```

Figure 5. Data to predict.

C. Delay Classes

The next step is to define classes for the various possible delays. Figure 6 shows the result.

```
classes = data['delay']
print('Classes: ' + classes.sort_values(ascending=True).
      unique())
classes = len(classes.unique())
print('Number of output classes: ' + classes)
```

```
Classes: [ 0 15 30 45 ]
Number of output classes: 4
```

Figure 6. Output of classes and number of classes.

To predict whether a delay might occur, delays must be classified. For this, the column with the delays is separated from the input variables and stored separately in a normalized form.

```
col_name = 'delay'
le = LabelEncoder()
col = le.fit_transform(data[col_name])

data = data.drop([col_name], axis=1)
```

D. Handling of Non-Numerical Data

Categories with non-numerical values - like firm, carrier and arrival - need to be converted into numerical values using one-hot-encoding (ohe). This binary coding increases the number of input variables for the next step.

```
conv_ohe = ['firm', 'carrier','arrival']
data = pd.get_dummies(data, columns=conv_ohe, dtype=float)
```

E. Training and Test Data

The last preparatory step is to divide the data into training and test data. A ratio of 80:20 is usually applied, where 80% are used for training and 20% for testing the model.

```
train_data, test_data, train_col, test_col =
      train_test_split(data, col, test_size = 0.2,
      random_state = 42)
```

F. Structure of the Artificial Neural Network

Keras sequential model is used for multi-class classification. It consists of a single stack of layers sequentially connected to each other.

```
model = tf.keras.Sequential([
  tf.keras.layers.Input(shape=(data.shape[1],)),
  tf.keras.layers.Dense(32, activation=tf.nn.sigmoid),
  tf.keras.layers.Dense(64, activation=tf.nn.sigmoid),
  tf.keras.layers.Dense(classes, activation=tf.nn.softmax)
])
```

- The input layer acts as the entry point for the data into the neural network.
- Two fully connected hidden layers build the inner neuronal structure. The first layer contains 32 neurons, and the second has 64. Both use the sigmoid activation function to map the input values between 0 and 1.
- The output layer has as many neurons as there are classes in the classification task and the softmax activation function selects the class with the highest probability as the predicted output.

G. Configure the Learning Process

The `model.compile()` function configures the learning process. This step involves specifying optimizer, loss function and evaluation metrics. The following setup was used in this study:

```
model.compile(optimizer='adam', loss='
      sparse_categorical_crossentropy', metrics=['accuracy
      '])
```

The network is trained using the *Adaptive Moment Estimation* (Adam) optimizer, a widely used gradient-based optimization algorithm. Adam combines the benefits of momentum and adaptive learning rates by maintaining running estimates of both, the first (mean) and second (uncentered variance) moments of the gradients. This makes Adam well-suited for training deep neural networks on noisy or sparse data sets and helps ensure stable convergence.

Given that the delay classes are encoded as integers, the `sparse_categorical_crossentropy` loss function computes the cross-entropy loss between the integer-encoded true labels and the probability distributions predicted by the network's softmax output layer.

The model's performance during training and evaluation is measured using it's accuracy. The metric calculates the proportion of correctly predicted delay classes relative to the total number of predictions. Accuracy provides an intuitive measure of the model's ability to correctly classify future delay categories based on the input features.

H. Execution of the Training Process

The training process of the neural network is conducted using the `model.fit()` function, which initiates the learning phase. The following configuration is applied:

```
model.fit(train_data, train_col, epochs=60)
```

The model is trained with the input feature matrix (`train_data`) and the target vector `train_col` containing the integer-encoded delay categories.

The training runs for 80 epochs to optimize the model's internal weights, reduce classification error, and avoid underfitting. Over successive epochs, the model is expected to converge to a solution that generalizes well to unseen data, and accurately classifying future delay categories based on new feature inputs.

Figure 7 provides a summary of the model's performance during the final five training epochs.

```
Epoch 75/80
25/25 [==============================] - 0s 1ms/step - loss
      : 0.2887 - accuracy: 0.9488
Epoch 76/80
25/25 [==============================] - 0s 1ms/step - loss
      : 0.2816 - accuracy: 0.9563
Epoch 77/80
25/25 [==============================] - 0s 1ms/step - loss
      : 0.2733 - accuracy: 0.9638
Epoch 78/80
25/25 [==============================] - 0s 1ms/step - loss
      : 0.2634 - accuracy: 0.9762
Epoch 79/80
25/25 [==============================] - 0s 983us/step -
     loss: 0.2551 - accuracy: 0.9762
Epoch 80/80
25/25 [==============================] - 0s 1ms/step - loss
      : 0.2507 - accuracy: 0.9850
```

Figure 7. Output of the training process.

The final training epochs show a consistent decrease in categorical crossentropy loss and a steady increase in classification accuracy. By the end of training, a training accuracy of 98.50% was acieved, indicating effective learning and convergence.

I. Testing

To complete the training, the model was evaluated on a test data set to assess its generalization performance. This evaluation was conducted using the following configuration. Figure 8 depicts the result of the test process.

```
75 test_loss, test_acc = model.evaluate(test_data, test_col)
76 print('Test accuracy:', test_acc)
```

```
7/7 [==============================] - 0s 975us/step - loss
      : 0.2226 - accuracy: 0.9800
Test accuracy: 0.9800000190734863
```

Figure 8. Output of the test process.

`model.evaluate()` computes the final loss and classification accuracy on the test data. These metrics provide insight into how well the model performs on previously unseen samples and help determine whether the network has successfully generalized beyond the training set.

The test accuracy reflects the proportion of correctly predicted delay categories in the test data set. This step is crucial to validate that the high training accuracy observed during the final epochs is not a result of overfitting.

The model achieved a test accuracy of 98.00% with a corresponding test loss of 0.2226 which indicates that the model has successfully generalized beyond the training set.

J. Make a Prediction

To evaluate the usability of the trained model, a separate test data set (`delivery_test_ohe.csv`) was used.

```
79 path2 = "../Data/delivery_test4.csv"
80 data_to_predict = pd.read_csv(path2, delimiter=';')
81
82 print('Data to predict: ')
83 print(data_to_predict)
```

To verify the correct alignment of feature values with the expected schema, the test data sample used for predictions was printed to the console as shown in Figure 9.

```
Data to predict:
   total  weekday  season  transport-distance  firm_prod1
      firm_prod2  \
0     16        3       4                 500        0.0
             0.0

   firm_prod3  carrier_car1  carrier_car2  carrier_car3  \
0         1.0           0.0           1.0           0.0

   arrival-time_evening  arrival-time_midday  arrival-
      time_morning
0                   0.0                  1.0
         0.0
1/1 [==============================] - 0s 58ms/step
```

Figure 9. Output of the customised CSV file for the prediction

The test data set was manually created and aligned with the one-hot-encoding-training data. Now, it could be used to perform a prediction:

```
86 pred = model.predict(data_to_predict)
87
88 print('prediction in %: ', pred)
```

The neural network returns a probability distribution across all output classes, indicating the model's confidence in each potential delay-time category. Figure 10 shows the prediction for the given input:

```
prediction in %:  [[4.4574207e-03 3.2497539e-06 1.8211146e
      -01 8.1342787e-01]]
```

Figure 10. Output of the prediction in per cent.

The array contains four floating-point values that sum approximately to 1.0. Each value represents the predicted likelihood of the input belonging to the respective class, i.e., 0.45% for class 0, 0.0003% for class 1, 18.21% for class 2, and

81.34% for class 3 indicating that the order will be 45min late.

The highest probability value of the prediction array was determined using the **NumPy** function `argmax()` and the result is shown in Figure 11.

```
# Determination of the index of the largest number
num = np.argmax(pred)
print('Index value: ', num)
print('_' * 75)
```

```
Index value:  3
```

Figure 11. Output of the prediction as an index value.

To decode the index it must be mapped back to its original string using the inverse transformation of the `LabelEncoder`:

```
# Determination of the result as string
spec = le.inverse_transform([num])
print('Maximum expected delay time: ', spec, 'min')
print('_' * 75)
```

`le` refers to the previously fitted `LabelEncoder` instance, and `num` is the predicted class index obtained via `np.argmax(pred)`. `inverse_transform[num]` converts this numerical index back into its original categorical form. The result is shown in Figure 12.

```
Maximum expected delay time:  [45] min
```

Figure 12. Output of the prediction.

This completes the model inference pipeline, converting raw input data into a meaningful delay-time prediction.

The models decision is influenced by multiple factors in the input:

- Carrier `car2` was associated with a significantly higher probability of delay (30 or 45 minutes).
- The season value of 4 corresponds to winter with further potential delays.
- The arrival time of “midday“ which frequently co-occurred with `car2` deliveries and longer delays.
- A transport distance of 500 km which was also characteristic of `car2` deliveries in the synthetic data.

VIII. Regression

A regression model is implemented to predict continuous numerical values - specifically, the expected delivery delay in minutes. The foundation for this model is based on the previously developed classification pipeline. With minor modifications it can be adapted to perform regression tasks.

A. From Classification to Regression

Data preprocessing from the classification section remains unchanged. This includes label encoding, one-hot-encoding for categorical features, and splitting the data set into training and test sets. Only the configuration of the artificial neural network and the loss function require adjustments. While input and inner layers stay unchanged, the output layer is defined as a single neuron without an activation function in order to predict continuous values.

```
model = tf.keras.Sequential([
  tf.keras.layers.Input(shape=(data.shape[1],)),
  tf.keras.layers.Dense(32, activation=tf.nn.sigmoid),
  tf.keras.layers.Dense(64, activation=tf.nn.sigmoid),
  tf.keras.layers.Dense(1)
])
```

B. Configuration of the Learning Process

Also, the learning process uses the Adam optimizer for regression. Now, mean-absolute-error (mae) is chosen as loss function. It measures the absolute, average magnitude of the errors between predicted and actual values. mae is also used as the performance metric to monitor the accuracy:

```
model.compile(optimizer='adam',loss='mae',metrics=['mae'])
```

C. Training Process

The training process is almost initiated in the same way as for classification. Except the epoch number is raised to 80 for a better adjustment of internal weights for regression. The training performs as shown in Figure 13.

```
model.fit(train_data, train_col, epochs=80)
```

```
Epoch 75/80
25/25 ------------------ 0s 1ms/step - loss: 0.2938 - mae:
      0.2938
Epoch 76/80
25/25 ------------------ 0s 1ms/step - loss: 0.3143 - mae:
      0.3143
Epoch 77/80
25/25 ------------------ 0s 1ms/step - loss: 0.3102 - mae:
      0.3102
Epoch 78/80
25/25 ------------------ 0s 1ms/step - loss: 0.2562 - mae:
      0.2562
Epoch 79/80
25/25 ------------------ 0s 1ms/step - loss: 0.2742 - mae:
      0.2742
Epoch 80/80
25/25 ------------------ 0s 1ms/step - loss: 0.2950 - mae:
      0.2950
```

Figure 13. Output of the prediction as an index value.

These results indicate that the model achieved a relatively stable and low mae during the final training phase, suggesting effective convergence. The fluctuations in the mae values are minor and typical for this type of training.

D. Testing

The model's performance was evaluated on beforehand unseen test data. Figure 14 shows the result.

```
16 test_loss, test_mae = model.evaluate(test_data, test_col)
17 print('Test mae:', test_mae)
```

```
7/7 ------------------ 0s 4ms/step - loss: 0.2826 - mae:
      0.2826
Test mae: 0.2971033751964569
```

Figure 14. Output of the prediction as an index value.

For 7 batches, the output shows a loss of 0.2826. This indicates that the average absolute difference between predicted and actual delay times is approximately 0.2826 minutes per instance. Also the `test_mae` metric delivers a comparable value. The small deviation between the two values stems from internal rounding variances.

E. Make a Prediction

Once the hyperparameters for a regression have been adjusted and the model has been trained and tested, the AI is ready to make predictions.

```
20 path2 = "../Data/delivery_test4.csv"
21 data_to_predict = pd.read_csv(path2, delimiter=';')
22
23 print('Data to predict: ')
24 print(data_to_predict)
25
26 # Make a prediction
27 pred = model.predict(data_to_predict)
28
29 print('delay time in min: ', pred)
```

```
Data to predict:
   total  weekday  season  transport-distance  firm_prod1
      firm_prod2  \
0     16        3       4                 500         0.0
            0.0

   firm_prod3  forwarder_car1  forwarder_car2
      forwarder_car3  \
0         1.0             0.0             1.0
      0.0

   arrival-time_evening  arrival-time_midday  arrival-
      time_morning
0                   0.0                  1.0
         0.0
1/1 ---------------- 0s 46ms/step
_____________________________________________
delay time in min:  [[3.0888479]]
_____________________________________________
```

Figure 15. Output of the prediction in minutes.

IX. ML Extended Simulation

It is the aim to integrate the ML solution presented in the previous sections into the formerly introduced simulation environment as shown in Figure 16. After the trusted orders are entered for simulation, a new phase **refurbish** is executed which makes use of the ML model.

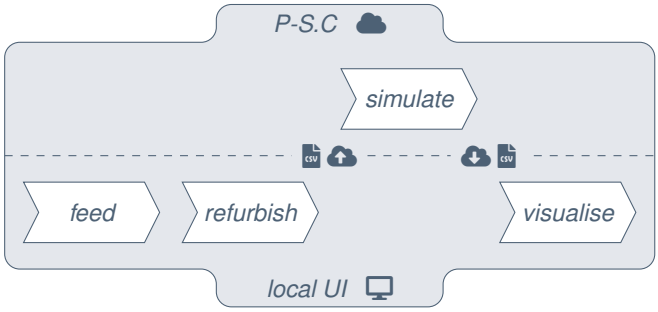


Figure 16. Simulation environment extended by ML.

With the goal to improve the ramp allocation, simulation is conducted in two phases. Figure 17 shows the improved planning approach.

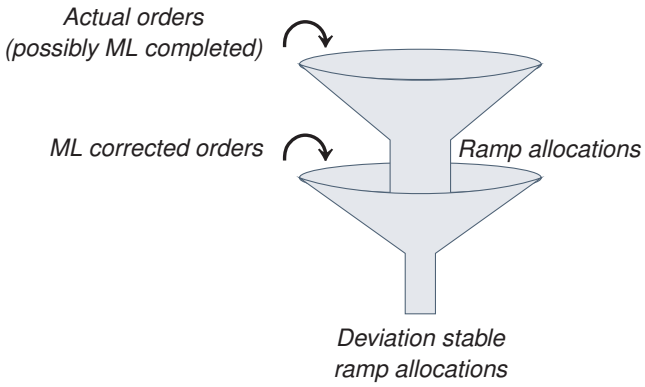


Figure 17. Two-Phase approach for ramp allocation.

A. Simulation of a Planned Schedule

In the first iteration, the orders are simulated for the case that all transports arrive as planned. If further orders are expected for the simulation period, the ML algorithm is able to generate plausible orders with respect to the known order history.

The simulation is conducted with the goal to find an optimal ramp allocation which reduces idle times for the shipping companies combined with a minimal labour utilisation. Different ramp allocations may lead to almost same results.

B. Simulation of a Planned Schedule plus (DNN) Delays

During the second phase, the orders are corrected with respect to the assumed deviations of transport times. The ramp allocation strategies found in the first phase are applied

to these adapted schedules and evaluated with respect to the optimisation criteria. These simulation results are used as a filter to find that allocation strategy which fits best to the original and the adapted orders.

C. Preliminary Work of Industry Partners

As mentioned before, the industry partner does not collect information needed for the machine learning approach. To prepare a data collection, the training data mentioned before is divided into three classes:

- **Available**: timestamp of the planned arrival; incoming vs. outgoing transports; amount of pallets
- **Not available but collectable**: the delay of transports; transportation distance in the sense of transports inside the industry park or nor; the shipping agent; the producer
- **Not available and not collectable**: in advance, the transportation distance in length

This classification reinforces the appraisal that the introduced approach can be conducted. It's application depends on the willingness of the industry partners.

X. Conclusion and Outlook

As initially stated, the goal of this research is to provide an environment to improve simulation in logistics. It uses machine learning based techniques to create schedules that are (mostly) indifferent to deviations from the original planning. Thus, a good schedule performs at least as good on the original data as the original schedule; but it performs as good as possible under differing scenarios.

A prototypical environment has been implemented, but the parts are not fully integrated yet. The reason for this is that the industry partner does not collect information of the deviation of planned and actual arriving time of the trucks. Also, other information that could help to train a neural net are not considered by the partner. The possibility to do so, has been explained above.

This represents the next important research step: check the approach with real-world data! Afterwards it makes sense to develop the integration further. The necessary implementation details have been uncovered in this article.

Even with this approach it is not possible to know which transports will be late in the future. But the developed scenario is more stable with respect to delays in delivery without degrading the initial plan. And for checking this plan, simulation is still needed. Purely statistical estimations are insufficient for processes in logistics.

References

[1] C. Simon, S. Haag, and N. G. Badurasvili, "Predictive AI To Feed Simulation", in *SIMUL 2024: The Sixteenth International Conference on Advances in System Simulation*, Venice (Italy), 2024, pp. 58–63.

[2] L. Zakfeld, C. Simon, M. Hladik, and S. Haag, "Real World Case Study To Teach Simulation", in *SIMUL 2023: The Fifteenth International Conference on Advances in System Simulation*, Valencia (Spain), 2023, pp. 19–24.

[3] Infraserv Logistics GmbH, *Overview hazardous substances warehouse*, https://www.infraserv-logistics.com/en/isl/news/news/ (last accessed 08.2024), 2023.

[4] C. Simon and S. Haag, "Pairing Finite Automata and Petri nets - Simulation of Processes in Logistics", in *ECMS 2024: 38th International ECMS Conference on Modelling and Simulation*, D. Grzonka, N. Rylko, and G. S. V. Mityushev, Eds., Krakow (Poland), 2024, pp. 474–480.

[5] C. Simon, S. Haag, and L. Zakfeld, "The Process-Simulation.Center", in *SIM-SC: Special Track at SIMUL 2022: The Fourteenth International Conference on Advances in System Simulation*, F. Herrmann, Ed., Lisbon (Portugal), 2022, pp. 74–77.

[6] T. Bergs *et al.*, "The Concept of Digital Twin and Digital Shadow in Manufacturing", *Procedia CIRP*, vol. 101, pp. 81–84, 2021, 9th CIRP Conference on High Performance Cutting, ISSN: 2212-8271.

[7] S. Russell and P. Norvig, *Artificial Intelligence - A Mordern Approach*, 4th ed. London: Pearson, 2021.

[8] P. Flach, *Machine Learning - The Art and Science of Algorithms that Make Sense of Data*, 9th ed. Cambridge: Cambridge University Press, 2012.

[9] J. Howard and S. Gugger, *Deep Learning for Coders with fastai & PyTorch*. Sebastopol: O'Reilly, 2020.

[10] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow*. Sebastopol: O'Reilly UK Ltd., 2019.

[11] C. Mattmann, Ed., *Machine Learning with TensorFlow*, 2nd ed. Shelter Island, NY: Manning, 2020.

[12] M. Karatas, *Development of AI applications (in German: Eigene KI-Anwendungen programmieren)*. Bonn: Rheinwerk Computing, 2024.

Data Integration, Querying and Reasoning over the Harmonized Survey on Households Living Standards Semantic Data Model

Marc Mfoutou Moukala
Faculty of Science and Technology
Marien NGOUABI University
Brazzaville, Congo
e-mail: moukmarc@yahoo.fr

Macaire Ngomo
Research and Innovation Department
CM IT CONSEIL
Romilly sur Seine, France
e-mail: macaire.ngomo@gmail.com

Régis Freguin Babindamana
Faculty of Science and Technology
Marien NGOUABI University
Brazzaville, Congo
e-mail: regis.babindamana@umng.cg

***Abstract*—In this paper, we build a Resource Description Framework (RDF)-based semantic data model of the Harmonized Survey on Households Living Standards (HSHLS), and propose an approach for data integration and querying and reasoning over the semantic data model built. We implement data consistency control rules, by combining RDF and Semantic Web Rule Language (SWRL). Based on SWRL rules, we then apply automated reasoning for the survey's data consistency control. We leverage the power of SPARQL to query information from the survey model and data, using Python programming language.**

Keywords-SPARQL queries; automated reasoning; semantic data model; Resource Description Framework data integration; Semantic Web Rule Language; data consistency control; Harmonized Survey on Households Living Standards.

I. INTRODUCTION

The work in [1] proposes a semantic data model of the Harmonized Survey on Households Living Standards (HSHLS) [2], which is a major household statistical survey conducted by French speaking countries, allowing public authorities to gather relevant information helping them to identify and solve daily living problems encountered by their population. The HSHLS survey consists of a set of modules or sections, each section dealing with a given theme. Among the topics addressed there are: the socio-demographic characteristics of households and their members as well as information on education, health and employment of household members. Information about each section is collected through a specific questionnaire administered using a computer application. The collected data is usually stored in a tabular structure in a relational database. After the data collection operations are executed, they are retrieved in Excel-type files for possible analyses. During data cleaning operations, discrepancies are often found between the methodology and the data actually collected, as some of the collected data do not comply with the conditions or rules defined in the methodology. Moreover, since the methodological information is not automated, data processing teams are often forced to manually consult the related documents; this sometimes causes delays in the data processing procedures. In the present work, we improve the semantic model built in [1], aimed at representing knowledge related to the survey under study in the way that facilitates the retrieval of methodological information. We also propose an approach to integrate survey data into the model built, and to query and make reasoning over the model. Doing so, the final purpose of the current work is both to document and disseminate knowledge related to this survey, and to facilitate data quality control, improving then the quality of the survey.

Our paper is structured as follows: In Section II, we define the concept of semantic data modeling and its advantages. In Section III, we present the literature review related to the semantic modeling in statistical surveys and highlight some limitations of the state of the art. Section IV presents the adopted methodology and tools. In Section V, we present our semantic data model and the implementation details. In Section VI, we propose and implement an approach to populate the built model with actual questions, and their respective conditions, constraints and dependencies. In Section VII, we prepare data consistency rules, using Semantic Web Rule Language (SWRL) [3], and apply automated reasoning over an example of data, based on those SWRL rules. In Section VIII, we present and discuss the results. Finally, we end with a conclusion along with an outline of potential future work, in Section IX.

II. SEMANTIC DATA MODELING

Data semantics is the meaning given to that data; it is its significance. It encompasses all the information that can be gathered about the data with respect to a specific objective and a particular reality. For example, regarding the data point **Age**, the following information can be inferred: Age is a property of a person, that defines their current lifespan, expressed in years, which is the mathematical difference between the year of birth and the current year, taking into account the date of the person's most recent birthday. This lifespan is an integer between 0

(minimum duration) and 120 (maximum duration). The year of birth and the current year are also properties of a person, of integer type. Semantic data modeling involves representing the data and their semantics as well as the relationships between them.

Semantic data models play a crucial role in the modeling of complex knowledge. They allow representing relationships and interactions between concepts in an accurate and structured way. These models are used in various application scenarios: representing relationships between concepts, encapsulating business knowledge, data search and analysis, integrating heterogeneous data sources, and supporting artificial intelligence and machine learning.

III. LITERATURE REVIEW

In the context of semantic modeling applied specifically to statistical survey data, the following research has strongly influenced our work.

The work in [4] addresses the challenge faced by users who need to write complex database queries to retrieve information, given their limited understanding of both the structural and semantic complexities of databases. It focuses on improving this process through the use of ontologies to facilitate better knowledge representation and interactive query generation.

In [5], the authors use an ontology-based approach to develop a semantic model for harmonizing and integrating population health data from heterogeneous sources. Following a presentation of the ontology literature, the authors of [5] identified key concepts and relationships between population health data. Then, they used this information to develop an XML schema-based semantic ontology to harmonize and integrate population health data from different sources (Excel, SQL Server and MongoDB) for early detection of COVID-19. The authors state that the model designed allows data to be inserted, updated and deleted without anomaly as the data mapping is based on schema and not on data. The authors also state that their method could be extended by creating ontologies in RDF/Turtle formats.

The work in [6] proposes an approach to improve the semantic interoperability of electronic health records using ontological management of domain ontology evolution. The researchers first developed a domain ontology representing concepts and relationships relevant to the domain of electronic health records. Then, they proposed methods to manage the evolution of this ontology over time, taking into account changes in the electronic health domain and new data requirements.

In [7], Nicholson et al. use an ontology-based approach to ensure a good level of data quality for cancer-related information in registries, in order to accurately compare indicators related to this disease on regional and national scale based on harmonized rules.

The work in [8] introduces a generic ontology designed to represent questionnaires in a machine-readable format. This ontology aims to enhance decision support systems and smart environments by facilitating automatic processing of questionnaire data, which has become more abundant and cost-effective due to mobile devices. It addresses the challenge of managing and reasoning about large volumes of collected information to gain deeper insights.

Considering the literature review, we notice that there is a great deal of similar work in semantic data modeling. However, to the best of our knowledge, there is no application of these research studies to the statistical harmonized housing surveys on household living standards. Access to methodological information is manual and the majority of data quality control is conducted manually, leading to significant delays in data processing. Therefore, our contribution lies in the application of this work in the context of documentation and popularization of knowledge relating to the HSHLS survey, and consists in proposing a semantic data model whose use would among other things, facilitate access to related knowledge and help speed up data processing.

IV. METHOD AND TOOLS

A. Methodology

The objective of this work is to design a semantic data model capable of formally and systematically representing the knowledge embedded in the HSHLS survey, with the aim of achieving the following goals: making the survey data and metadata interoperable, improving efficiency across the survey lifecycle, facilitating advanced and flexible querying, and supporting data quality control.

The HSHLS survey involves complex and multi-level data, and managing that survey effectively requires formal structures that can support automated processes, ensure conceptual consistency, and validate the integrity of collected and processed data. To meet these challenges, the literature review led us to adopt an ontology-based semantic modeling approach to represent and structure knowledge semantically. This choice is justified by the fact that:

- Ontologies enable complex logical relationships between concepts to be defined formally. Axioms and rules can be used to express logical conditions of dependency between survey questions such as IF-THEN conditions, validation constraints, hierarchical relationships, etc.
- Ontologies provide a standardized, shared data model, promoting interoperability between different systems and enabling easier data integration. This can be particularly useful in a survey context, where data needs to be collected, stored and analyzed in a consistent and standardized way.
- Ontologies enable the complexity of dependencies between survey questions to be managed in a structured way. Concepts can be organized into classes and sub-classes and properties and restrictions can be defined, making it easier to manage and understand the relationships between different questions.

- Ontologies provide a solid basis for managing the evolution and maintenance of the semantic data model. Concepts and relationships can be easily added, modified or deleted without compromising model consistency and compatibility.

B. Tools

Although there are many works available in the literature that use other representation ways, to build our semantic model, represent concepts and their relationships, we use the Resource Description Framework (RDF) [9] and RDF Schema (RDFS) [10] along with the Web Ontology Language (OWL) [11]. The reasoning part is achieved using Semantic Web Rule Language (SWRL).

RDF is a framework standardized by the World Wide Web Consortium (W3C) to represent information on the Web in a structured and interoperable way. It provides a simple data model based on subject-predicate-object assertions, also known as RDF triples. Each triple describes a relationship between two resources.

RDF Schema (RDFS) is an extension of RDF that provides a vocabulary for describing schemas and ontologies. This enables the definition of classes, properties and relationships between RDF resources.

To learn about the various concepts related to the survey under study as well as the relationships between the data, this research relies on methodological documents including household questionnaires, interviewer manuals and survey data dictionaries.

Our semantic model is built using RDF and RDF Schema. We propose to use metamodeling techniques to implement the models needed to manipulate the elements of the Harmonized Survey on Households Living Standards questionnaire and the survey data consistency control. The following section presents our semantic model.

V. MODELIZATION AND IMPLEMENTATION

The original concepts of the survey under study are in French, since this survey concerns French-speaking countries. However, for illustration purposes, in this work, we provide an English version of those concepts to allow a large audience to easily understand our work.

A. Definitions of Key Concepts

1) Harmonized Survey on Households Living Standards(HSHLS): HSHLS is the main harmonized statistical survey conducted by French-speaking countries in West and Central Africa to capture household living conditions.

2) Section: The HSHLS survey is composed of sections. A section is named according to the topic addressed: Socio-demographic characteristics, Education, Health, etc.

3) Questionnaire: Every section has a single questionnaire. A questionnaire captures the main information of surveyed households related to a given section.

4) Question: A questionnaire is composed of questions.

5) Household: This is the main statistical unit on which information are gathered. We distinguish two kinds of households: ordinary households and collective households. An ordinary household is a group of people, whether related or not, who usually live in the same dwelling, pool their resources, share their meals, and recognize the authority of the same person as the head of the household. Households that are not ordinary are referred to as collective households. These are people or groups of people living in collective housing (such as military barracks, boarding schools, hospitals, etc.). This survey only concerns ordinary households. An ordinary household consists either of a single person (for example, a student who rents a room alone) or several people. In the latter case, the household usually consists of a spouse, their husband/wife, and their children, with or without other dependents (family members, friends, etc.). An ordinary household can also be made up of people who live together and have no familial ties. In all the rest, the concept of household refers to an ordinary household.

6) Household member: A person belonging to a particular household. A household member is a person who usually resides in the household. An individual usually resides in the household in two situations: he/she has lived in the household for at least 6 months or has arrived in the household less than 6 months ago but with the intention of staying for at least 6 months.

B. Presentation of the HSHLS Semantic Model

Our model consists of a resource class HSHLS representing HSHLS surveys, an instance of the rdfs:Class class of the RDF model. A survey is made up of sections. A section contains a questionnaire. A questionnaire is of a certain type, depending on the information collected. This may be information characterizing household members or common household characteristics. These characteristics are variables represented by questions. Each question is an instance of the rdfs:Class class in our model, and concerns a household member or a household as a whole. A household member belongs to a unique household, and has answers to the survey questions. A household member can have answers to all or some questions, depending on their characteristics and the eligibility criteria (for example, a minimum age threshold for a surveyed to answer to questions on Education). An answer is related (refers) to both a household member and a question, and is of a certain type (integer, float, string) depending on the nature of the expected response. Each question is linked to a questionnaire. A question may depend on other questions and may have constraints, which are acceptable values of its answers.

The general model includes the class declaration model (Figure 1) and the property declaration model (Figure 2). More details of the semantic model are given in Appendix A.

In the class declaration, the declaration C rdf:type C' means that the class C is an instance of the class C'. For example, hshls:HSHLS rdf:type rdfs:Class states that the HSHLS is an instance of rdfs:Class. In the property declaration, a triple of the form: P rdfs:domain C declares that P is an instance of the rdf:Property class, that C is an instance of the rdfs:Class class, and that the resources indicated by the subjects of triplets whose predicate is P are instances of the C class. This implies that hshls:hasQuestion rdfs:domain hshls:Questionnaire states that hasQuestion is an instance of rdf:Property class, Questionnaire is an instance of the rdfs:Class class, and that the resources indicated by the subjects whose predicate is hasQuestion are instances of the class Questionnaire. The triple P rdfs:range C means that P is an instance of the rdf:Property class, that C is an instance of the rdfs:Class class, and that the resources indicated by the objects in the triple whose predicate is P are instances of the C class. In this case, hshls:hasQuestion rdfs:range hshls:Question means that hasQuestion is an instance of the rdf:Property class, Question is an instance of the rdfs:Class class, and that the resources indicated by the objects in the triple whose predicate is hasQuestion are instances of the Question class. The following code illustrates the class declaration model, serialized in Turtle format, and its (simplified) graph representation (Figure 1).

```
hshls:HSHLS rdf:type rdfs:Class ;
    rdfs:label "Harmonized Survey on
Households Living Standards" .
hshls:Section rdf:type rdfs:Class ;
            rdfs:label "A section or
module of the survey" .
hshls:Questionnaire rdf:type
rdfs:Class .
hshls:Question rdf:type rdfs:Class ;
    rdfs:label "A question of the
survey" .
hshls:Constraint rdf:type rdfs:Class ;
          rdfs:label "Constraint on
the question" .
hshls:Household rdf:type rdfs:Class
;
    rdfs:label "A household surveyed"
.
hshls:Household_Member rdf:type
rdfs:Class ;
    rdfs:label "A household member
surveyed" .
hshls:Answer rdf:type rdfs:Class ;
    rdfs:label "A household member's
answer related to a question" .
```

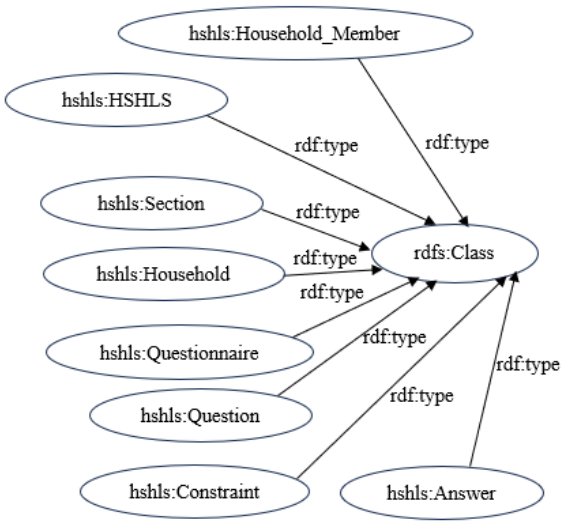


Figure 1: HSHLS RDFS model graph with class declaration.

Figure 1 represents 8 class declarations, all being instances of rdfs:Class.

Figure 2 illustrates the property declaration model, with the properties written in bold. For space-saving purposes, not all properties are represented in the figure.

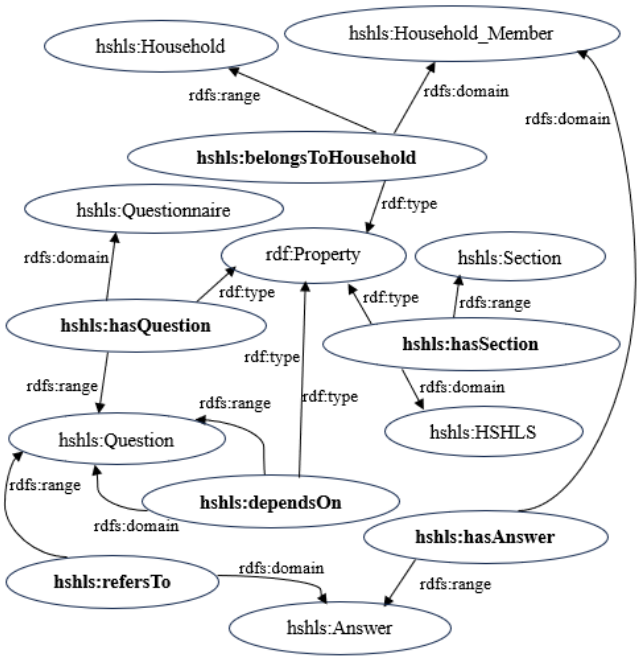


Figure 2: HSHLS RDFS model graph with property declaration

C. HSHLS Model Specialization

To serve as a proof of concept, we propose a specialization of our model, considering personalized methodological information used during the first edition of this survey in Congo. The overall survey has 21 sections. In this article, we do consider a particular section: the Education section. The specialization is as follows:

1) HSHLS ontology with actual questions: Figure 3 illustrates the map of HSHLS ontology with actual questions, for the section which captures education's information, codified S02. The education information concerns household members of three (3) years old or above. So, the entry in this questionnaire depends on the response to the question on the age of the corresponding household member surveyed, here codified S02Q_Age. The question S02Q1a captures whether the surveyed household member can read a little text written in French. S02Q1a depends on the question S02Q_Age: if the value of the answer to the question S02QAge by the concerned household member is less than a given minimum (9, in this case), the question S02Q1a must not be asked, so the value of the answer related to that question must be empty. The question S02Q03, which also depends on the question S02Q_Age, captures whether the surveyed household member is currently attending or have attended a formal school. The question S02Q04 captures the reason why the surveyed household member has never attended a formal school. S02Q04 depends on the response to the question S02Q03 which can be "Yes" or "No, never attended" meaning that the concerned surveyed has never attended a formal school. S02Q04 is asked only if the response to S02Q03 takes the second valid value. A formal school (public or private) is a place of learning where the programs offered are organized and structured (primary school, high school, university, etc.), and formal learning typically leads to the validation and awarding of a diploma.

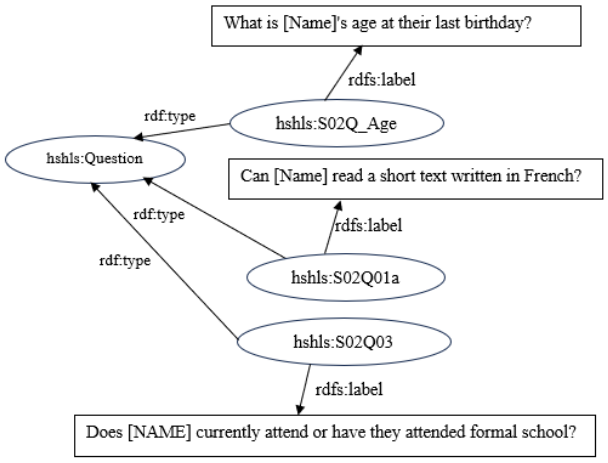


Figure 3: The HSHLS RDFS Ontology with actual questions.

In Figure 3, triples X rdf:type hshls:Question (where X is one of hshls:S02Q_Age, hshls:S02Q01a and hshls:S02Q03) mean that X is an instance of hshls:Question, in other words, X is a question. Triples X rdfs:label y mean that the question X has as label y.

2) HSHLS constraints and dependency specification: Figure 4 gives an illustration of actual questions and their constraints and dependencies. A constraint in our model represents valid values of a question. In Figure 4, the triple hshls:ConstraintS02Q01a rdf:type hshls:Constraint declares that hshls:ConstraintS02Q01a is a constraint. The property hshls:hasConstraint links a question to its constraint. Both a question and a constraint being instances of resources of rdfs:Class, the property hshls:hasConstraint is an object property. The data property hshls:validValues specifies the valid (possible) values of the related constraint. In the figure, the triples hshls:ConstraintS02Q01a hshls:validValues "Yes" and hshls:ConstraintS02Q01a hshls:validValues "No" ("Yes" and "No" being literal values) specify that the valid values of the constraint hshls:ConstraintS02Q01a are "Yes" and "No". That means that the response to the related question (S02Q01a) must be "Yes" or "No". In the figure, the question S02Q01a depends on the question S02Q_Age and the dependency condition is that the value of the answer to S02Q_Age (which precedes S02Q01a) must be greater or equal to 9. That means, to ask the question S02Q01a, the concerned surveyed must be 9 years old or above. If this condition is not satisfied, then the answer to S02Q01a must be empty for the concerned surveyed.

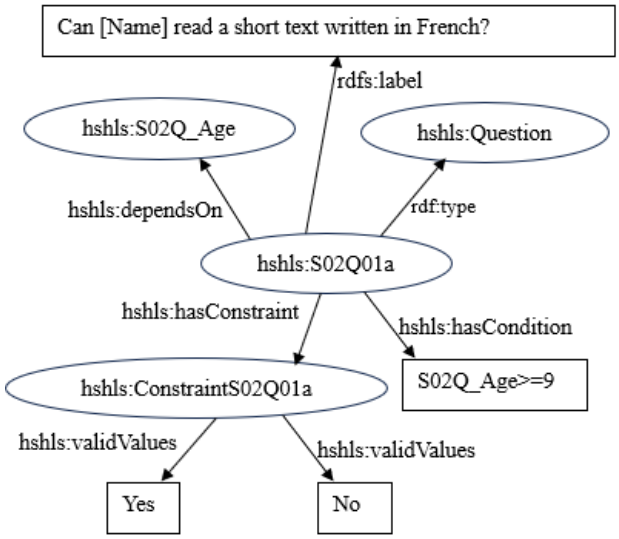


Figure 4: HSHLS ontology constraints and dependency specification.

To enhance the clarity of constraint meanings and enable seamless navigation within the model, we chose using a clear and intuitive coding scheme for each

constraint, as outlined hereafter: ConstraintQuestionName where QuestionName is the name of the related question. For example, hshls:ConstraintS02Q01a in Figure 4 denotes the constraint related to the question S02Q01a.

In the above, we built a semantic data model to represent the Harmonized Survey on Households Living Standards (HSHLS) methodological information in a human and computer readable format. The proposed model is built using the Python RDFLib package [12], in a Jupyter notebook environment. The model is saved in Turtle (ttl) format and can be exploited as an RDF graph. To make it possible to get access to the model in a persistent way, we defined an International Resource Identifier (IRI) for the model. We also developed an HTML ontology documentation file using PyLODE [13]. We created a public Github repository [14] and saved the rdf graph and its HTML documentation in it.

In the following sections, we propose and implement a way to populate the model and to make reasoning based on the model. In this work, the reasoning system is intended for data consistency control, in order to solve the problem of discrepancies which are often found (during data cleaning operations) between the methodology and the actual data collected. We propose the reasoning system to make it possible to regularly detect the inconsistencies in data during data collection operations, improving then the efficiency of the survey operations.

The rest of this paper is structured as follows: in Section VI, we propose and implement an approach to automatedly populate the semantic data model with actual questions, and their respective conditions, constraints and dependencies. Section VII presents the automated reasoning mechanism proposed for the survey's data consistency control, through SWRL rules. This section ends with a presentation of some SPARQL queries over the data model built. In Section VIII, we present and discuss the results. Finally, we end with a conclusion along with an outline of potential future work, in Section IX.

VI. HSHLS RDFS ONTOLOGY AND DATA INTEGRATION

In this section, we propose and implement an approach to integrate model's ontology and data in an automated way. The aim of the ontology integration is to map different instances of the model (questions, labels, constraints, dependencies, conditions) from different survey versions into a formal RDF structure that conforms to our existing model. The data integration part proposes a way to integrate actual data from the survey into the RDFS model.

1) HSHLS RDFS ontology integration: In our approach, we prepare the instances of the model in an Excel-type file, and we integrate those instances into the model, using Python.

Figure 5 illustrates the implemented process. We first load the Turtle RDF model and the Excel file containing questions and their description (Question ID, label, constraint, dependency, condition). Next, URIs are generated for every element of each iterated row, and triples are added to the RDF graph based on generated URI. Finaly, the updated RDF graph is saved.

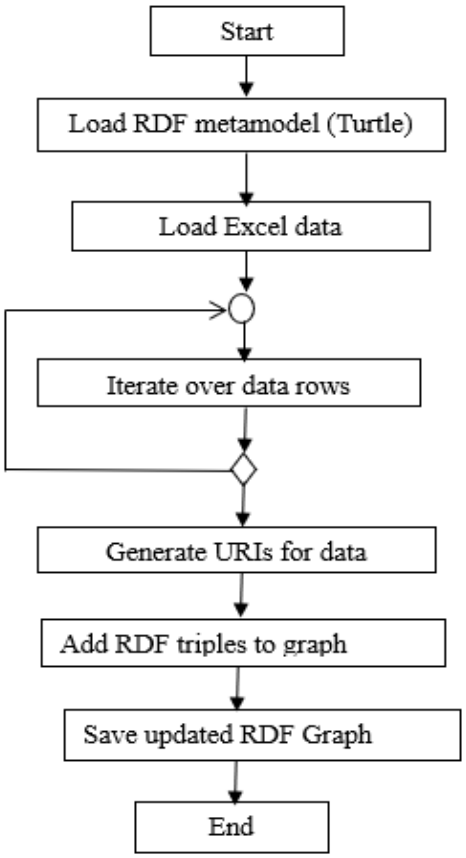


Figure 5: Diagram for populating the HSHLS RDF model.

A pseudo-code representation of the presented process is detailed as follows:

```
Begin
    Initialize an empty graph ;
    Parse the RDF Turtle file into the
graph ;
    Load the Excel file containing the
data (questions, constraints,
conditions) ;
    For each row in the DataFrame:
        Generate URIs for question and
constraint;
        Construct the URI for the
question using the "Question ID"
column ;
        Construct the URI for the
constraint using the "Constraint"
column ;
```

```
        Add a triple : (question URI,
RDF.type, Question class) ;
        Add a triple: (question URI,
hasConstraint, constraint URI) ;
Add a triple : (question URI,
dependsOn, value from "Dependency"
column) ;
        Add a triple : (question URI,
hasCondition, value from "Condition"
column) ;
        Add a triple : (constraint
URI, RDF.type, Constraint class) ;
        Add a triple: (constraint URI,
validValues, values from "Constraint"
column) ;
    Serialize the updated graph to a
new Turtle file and save it.
End
```

Table I and Table II (which actually constitute the same table but separated for space-saving purposes) illustrate the shape of the Excel file with questions to insert automatedly in the survey semantic data model.

TABLE I. HSHLS SURVEY QUESTIONS AND THEIR LABELS AND CONSTRAINTS

Question ID	label	Constraint
S02Q01a	Can [Name] read a short text written in French?	Yes;No
S02Q01b	Can [Name] read a short text written in Lingala?	Yes;No

TABLE II. HSHLS SURVEY QUESTIONS AND THEIR CONDITIONS AND DEPENDENCIES

Question ID	Dependency	Condition
S02Q01a	S02Q_Age	S02Q_Age>=9
S02Q01b	S02Q_Age	S02Q_Age>=9

In Table I and Table II, **Question ID** is the codification of the question. The column **label** represents the way to ask the given question to the respondent or the surveyed. The column **Constraint** represents valid values of the related questions. The column **Dependency** specifies the question(s) on which the given question depends, and the dependency condition is given in the column **Condition**. In other words, the given question may remain unanswered, depending on the answers to the question(s) on which it is contingent. In the case illustrated in the table, the decision to ask question S02Q01a depends on the response given to question S02Q_Age, and the dependency condition is that S02Q_Age must be greater or equal to 9.

The semantic data model with concepts inserted automatedly following the described process for questions presented in Table I and Table II is presented as follows:

```
  hshls:ConstraintS02Q01a
  a hshls:Constraint ;
      hshls:validValues
"No"^^xsd:string,
          "Yes"^^xsd:string .
  hshls:ConstraintS02Q01b
  a hshls:Constraint ;
      hshls:validValues
"No"^^xsd:string,
          "Yes"^^xsd:string .
  hshls:S02Q01a a hshls:Question ;
      rdfs:label " Can [Name] read a
short   text   written   in   French?
"^^xsd:string ;
      hshls:dependsOn "S02Q_Age" ;
      hshls:hasCondition  "S02Q_Age>=9"
;
      hshls:hasConstraint
hshls:ConstraintS02Q01a .
  hshls:S02Q01b a hshls:Question ;
      rdfs:label " Can [Name] read a
short   text   written   in   Lingala
?"^^xsd:string ;
      hshls:dependsOn "S02Q_Age" ;
      hshls:hasCondition  "S02Q_Age>=9"
;
      hshls:hasConstraint
hshls:ConstraintS02Q01b .
```

2) HSHLS RDFS data integration: In this part, we propose a way to retrieve actual data from an input data file, and insert them into the model built. Here's an explanation of each step involved in this workflow:

- Initialize RDF Graph: This step involves creating an empty RDF graph, which will store all the triples (subject-predicate-object relationships) for the data.
- Load existing RDF model: Load the RDF survey model to ensure that the new data is added to the pre-defined structure.
- Load survey data from Excel file: The Pandas library is used to read the survey data from an Excel file into a DataFrame. The survey data, which includes household details, household members information, and answers to various questions, is stored in a structured tabular form (rows and columns). This makes it easier to iterate over and extract specific information for RDF generation.
- Iterate through each row in the DataFrame: The program iterates through each row in the

DataFrame, processing the data for one household member at a time. Each row corresponds to a specific household member and its associated answers.

- Extract household information: For each row, key details related to the household are extracted, such as: department number, which is the geographic region of the household; the cluster number; the household number (a unique identifier for the household within the cluster), and the wave number which indicates the survey wave or time period the data is from.
- Create and add household URI to RDF: Using the extracted household information, a unique URI is generated for each household, following a naming pattern such as: household_departementNumber_clusterNumber_householdNumber_waveNumber. This URI is used as the subject for all triples related to the household. this makes it possible to consistently reference and link the household's data (members, answers) within the RDF graph.
- Create and add household member information: A household member is given an order number inside the concerned household. But that number is not unique across the entire survey. So, to uniquely identify a household member across the entire survey and ensure proper linking in the RDF model, we propose a combination of the household identifier and the household member order number. Then, for each member of the household, the program generates a unique URI. The program adds RDF triples for each member, including they details. These triples represent key attributes of the household member in the RDF graph. A triple is also added to link the household member to their corresponding household using the predicate *belongsToHousehold*. This relationship ties each member to the household, creating a clear connection between the two entities in the RDF graph.
- Add answers for survey questions: The program iterates over the columns of the DataFrame that represent survey questions (S02Q_Age, S02Q01a, etc.). For each question, the program checks if there's a non-null value. This ensures that only non-empty answers are processed, excluding any empty or missing data. For each valid answer, a unique URI for the answer is created, incorporating details of the household member identifier and the related question. RDF triples are added to the graph, associating the answer URI with the answer's value (a numeric or textual response). The answer is also linked to the relevant question URI, and the member is linked to the answer using the predicate *hasAnswer*. This ensures that each survey response is recorded as a separate RDF entity, linked to both the question and the household member who provided the answer.
- Serialize and save the updated RDF Graph: After all rows are processed and the RDF graph is populated with all the data, the program serializes the graph into a Turtle (.ttl) file. This allows the RDF graph to be saved in a standard format that can be shared, queried, or used by other applications. At the end, the survey turtle file will contain the RDF representation of the entire survey.

Table III and Table IV illustrate an example of survey data from households in the 11th department, from three distinct clusters (the names of some columns are shortened and some columns are not represented for space-saving purposes).

TABLE III. EXAMPLE OF HSHLS SURVEY DATA

departement	cluster	householdNumber	waveNumber
11	516	4	2
11	516	4	2
11	516	4	2
11	320	2	2
11	324	4	2

TABLE IV. EXAMPLE OF HSHLS SURVEY DATA (CONTINUE)

OrderNumber	Name	Sex	S02Q_Age
1	Mouk	M	51
2	Marc	F	32
3	Annan	F	2
1	Lotté	M	56
2	Bruno	M	8

In Table III and Table IV, each row represents the responses of a respondent or a surveyed to the questions for a given section. Columns **departement**, **cluster**, **householdNumber**, **wavenumber** and **OrderNumber** compose identification information of the surveyed. An illustration of the result of data integrated is given as follows:

```
hshls:householdmember_11_320_2_2_1
a hshls:Household_Member ;
    hshls:Name "Lotté"^^xsd:string ;
    hshls:Sex "M"^^xsd:string ;
    hshls:belongsToHousehold
hshls:household_11_320_2_2 ;
```

```
    hshls:hasAnswer
hshls:answer_11_320_2_2_1_S02Q01a,
hshls:answer_11_320_2_2_1_S02Q03,
hshls:answer_11_320_2_2_1_S02Q_Age .
hshls:answer_11_320_2_2_1_S02Q01a
a hshls:Answer ;
    hshls:hasValue "Yes" ;
    hshls:refersTo hshls:S02Q01a .
hshls:answer_11_320_2_2_1_S02Q03
a hshls:Answer ;
    hshls:hasValue "Yes" ;
    hshls:refersTo hshls:S02Q03 .
hshls:answer_11_320_2_2_1_S02Q_Age
a hshls:Answer ;
    hshls:hasValue 56 ;
    hshls:refersTo hshls:S02Q_Age .
hshls:household_11_320_2_2
a hshls:Household ;
    hshls:clusternumber
"320"^^xsd:string ;
    hshls:departementNumber
"11"^^xsd:string ;
    hshls:householdNumber
"2"^^xsd:string ;
    hshls:waveNumber "2"^^xsd:string .
```

VII. AUTOMATED REASONING

Automated reasoning is a branch of artificial intelligence that focuses on the development of algorithms and software tools that allow machines to deduce new knowledge or validate logical arguments without human intervention. In our work, we apply automated reasoning for data consistency verification. We first build a set of data consistency rules, using SWRL, next, we import actual data into Protégé software and apply those SWRL rules against actual data. The result of the reasoning is saved into an RDF-OWL Turtle file for querying purposes. In the following, we first give an overview of SWRL mechanisms and the software (Protégé) used in our work for reasoning, before presenting SWRL rules defined on an example of survey data, and the inferred axioms from the described reasoning software.

A. Overview of Semantic Web Rule Language

Semantic Web Rule Language (SWRL) is a rule-based framework that extends Web Ontology Language (OWL) and RDF with logic-based rules, enabling the Semantic Web to infer new knowledge from existing data. Combining OWL and Horn Logic, SWRL allows for the definition of logical rules that can infer new knowledge from existing data, facilitating dynamic and intelligent web-based systems. SWRL's integration with RDF enables it to operate over structured data, making it a critical component for knowledge representation, inference, and automated reasoning. SWRL is essentially an intersection of first-order logic and description logic. It allows for a more dynamic and complex set of reasoning capabilities over ontologies. By utilizing rules, SWRL supports the reasoning of facts and the automatic generation of inferences based on the provided input. SWRL's syntax is based on Horn Logic, where each rule is an implication that takes the form of an "if-then" statement. A typical rule has an antecedent (the "if" part), which represents conditions or facts that must hold, and a consequent (the "then" part), which defines the new fact that will be inferred if the conditions are true. The general structure of a SWRL rule is: **Antecedent -> Consequent**, for example:

```
Person(?p)^hasAge(?p,?a)^
swrlb:greaterThan(?a, 18) -> Adult(?p).
```

This rule states that if a Person has an age greater than 18, they should be classified as an Adult. SWRL rules can also be expressed in the form of **body** and **head**, where body consists of a set of conditions or premises that must be satisfied for the rule to apply (equivalent to Antecedent), and head contains the conclusion that follows when the conditions in the body are met (equivalent to Consequent). SWRL rules can involve complex logic, including data type comparisons, existential quantification, and object property relations. Once SWRL rules are defined, they can be processed by reasoning engines such as Pellet [15], HermiT [16], or FaCT++ [17]. These reasoning engines are capable of interpreting the rules and performing inference over the knowledge base represented by the OWL ontology. When the reasoning engine processes an ontology with SWRL rules, it applies the rules to infer new facts or detect contradictions. This process makes SWRL especially valuable for tasks such as data validation, automated decision-making, and semantic search.

B. Overview of Protégé Software

In our work, we use Protégé (software, version 5) to implement the rules and apply reasoning over the RDFS-OWL data model built. Protégé ([18], [19]) is a free, open-source platform (W3C standards compliant) that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. Protégé is offered in two formats: Protégé Desktop and WebProtégé. Protégé Desktop (used in our work) is a feature rich ontology editing environment with full support for the OWL 2 Web Ontology Language, and direct in-memory connections to description logic reasoners like HermiT and Pellet. Protégé Desktop supports creation and editing of one or more ontologies in a single workspace via a completely customizable user interface. Visualization tools allow for interactive navigation of ontology relationships. Advanced explanation support aids in tracking down inconsistencies. Refactor operations available including ontology merging, moving axioms between ontologies, rename of multiple entities, and more. WebProtégé is an ontology development environment for the Web that makes it easy to create, upload, modify, and share ontologies for collaborative viewing and editing.

C. Automated Reasoning over the Harmonized Survey on Households Living Standards Data Model

In the survey under study, there are some dependencies between questions, and those dependencies specify some eligibility conditions for surveyed household members in respect to some questions. In others words, not all questions are responded by all surveyed, depending on the age range, sex, and others characteristics. For example, a surveyed of less than three years old cannot attend a school. For that constraint, an eligibility condition is defined such that: if S02Q_Age < 3, then skip all questions related to the education of the surveyed. In the same way, if a surveyed answered "Yes" to the question S02Q03 asking if he/she is attending or has ever attended a formal school, it should be inconsistent to ask the question S02Q04 to know the reason why the surveyed didn't attend a formal school. If the answer to S02Q03 is "No, never attended", we need to make sure that the reason of non-attendance is captured, and we need to skip all school attendance related questions such as asking the highest school level achieved by the surveyed and others. So, in regard to all the precedent, we chose to translate those constraints into rules aiming to facilitate reasoning. The code below illustrates those rules, built using SWRL and executed using Protégé:

```
hshls:Household_Member(?HMember)^
hshls:hasAnswer(?HMember,
?answer)^hshls:refersTo(?answer,
hshls:S02Q_Age)^hshls:hasValue(?answer,
?value)^swrlb:lessThan(?value,3)^
hshls:hasAnswer(?HMember,
?answer2)^hshls:refersTo(?answer2,
hshls:S02Q03)->
hshls:Inconsistent(?HMember)^
hshls:HasInconsistency(?HMember,
"Inconsistent data. The age of this
individual is less than 3, therefore he
or she must not answer to questions on
Education. Please double check and
correct the information accordingly.")
```

The rest of the rules can be found in Appendix B.

After setting rules and executing them in Protégé, we get inferred axioms with potential inconsistencies pinpointed.

D. Querying on the Harmonized Survey on Households Living Standards Semantic Data Model

In this subsection, we present the feasibility of doing some queries for information retrieval from the survey ontology and data. We use SPARQL for that purpose. We first give a brief overview of SPARQL concepts and mechanisms before presenting some queries as a proof of concept.

1) An overview of SPARQL: SPARQL [20] is a query language designed specifically for querying RDF data. RDF structures data in triples: subject-predicate-object, where the subject represents the entity, the predicate represents the property, and the object represents the value of that property.

Below are some types of queries included in SPARQL:

- SELECT: The fundamental query in SPARQL is the SELECT query, which retrieves specific data from RDF stores based on pattern matching. A typical SPARQL query looks like this:

```
SELECT ?name WHERE {
?person rdf:type foaf:Person .
?person foaf:name ?name .}
```

In this example, the query selects the names of all individuals classified as foaf:Person.

- CONSTRUCT: Generates a new RDF graph based on the results of a query.
- ASK: Returns a boolean answer indicating whether a given query pattern exists in the dataset.
- DESCRIBE: Provides an RDF graph that describes the resources identified by the query.
- FILTER expressions: Allow for complex logical and arithmetic operations to restrict query results.

2) SPARQL queries for HSHLS survey methodological information retrieval: Following are some queries for illustration:

- List questions and their dependencies: Here is a query to list questions and their dependencies:

```
from rdflib.plugins.sparql import
prepareQuery
query_onto_depends = prepareQuery('''
PREFIX
hshls: <http://w3id.org/HshlsOnto/>
PREFIX
owl: <http://www.w3.org/2002/07/owl#>
PREFIX
rdf:  <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>
PREFIX
rdfs:  <http://www.w3.org/2000/01/rdf-
schema#>
PREFIX
xsd:
<http://www.w3.org/2001/XMLSchema#>
SELECT ?question ?dependency
WHERE {
?question rdf:type hshls:Question .
?question  hshls:dependsOn  ?dependency
.
}
 ''')
```

Execute the query and check the results:

```
results_onto_depends=
graph.query(query_onto_depends)
for row in results_onto_depends:
    print(row)
```

The result is as follows:

```
(rdflib.term.URIRef('http://w3id.org/H
shlsOnto/S02Q01a'),
rdflib.term.URIRef('http://w3id.org/Hs
hlsOnto/S02Q_Age'))
(rdflib.term.URIRef('http://w3id.org/H
shlsOnto/S02Q03'),
rdflib.term.URIRef('http://w3id.org/Hs
hlsOnto/S02Q_Age'))
(rdflib.term.URIRef('http://w3id.org/H
shlsOnto/S02Q04'),
rdflib.term.URIRef('http://w3id.org/Hs
hlsOnto/S02Q03'))
```

The interpretation of this result is the following: the question S02Q01a depends on S02Q_Age, S02Q03 depends on S02Q_Age, and S02Q04 depends on S02Q03.

The result can be further formated in a more human-readable way.

- List questions with their dependency conditions:

```
query_onto_condition = prepareQuery('
'
 PREFIX hshls: <http://w3id.org/HshlsO
nto/>
 PREFIX owl: <http://www.w3.org/2002/0
7/owl#>
PREFIX rdf: <http://www.w3.org/1999/02
/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/0
1/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XM
LSchema#>
SELECT ?question ?dependency_condition
WHERE {
        ?question rdf:type hshls:Quest
ion .
        ?question hshls:hasCondition ?
dependency_condition .
    }
''')
```

Execute the query and check the results:

```
results_onto_condition = graph.query(q
uery_onto_condition)
for row in results_onto_condition:
    print(row)
```

The result is as follows:

```
(rdflib.term.URIRef('http://w3id.org/H
shlsOnto/S02Q01a'), rdflib.term.Litera
l('( S02Q_Age >= 9)'))
(rdflib.term.URIRef('http://w3id.org/H
shlsOnto/S02Q03'),   rdflib.term.Liter
al('( S02Q_Age >= 3)'))
(rdflib.term.URIRef('http://w3id.org/H
shlsOnto/S02Q04'),   rdflib.term.Liter
al("(S02Q03 = No, never attended)"))
```

3) SPARQL queries for data consistency information retrieval: During the reasoning process, we built a class for inconsistent data, and a property dedicated to pinpointing the inconsistency. The Turtle RDF below is an example of a household member information.

```
hshls:householdmember_11_516_4_2_3
a hshls:Household_Member ;
    hshls:Name "Annan"^^xsd:string ;
    hshls:Sex "F"^^xsd:string ;
    hshls:belongsToHousehold
hshls:household_11_516_4_2 ;
    hshls:hasAnswer
hshls:answer_11_516_4_2_3_S02Q01a,
hshls:answer_11_516_4_2_3_S02Q_Age .
```

Below is one of SPARQL requests prepared and tested:

```
# Select individuals who are household members and are
marked with the inconsistency class
SELECT ?member ?label  WHERE {?member
rdf:type hshls:Household_Member ;
rdf:type hshls:Inconsistent ;
hshls:HasInconsistency ?label .}
```

In this request, we retrieve information about household members whose some information contains inconsistencies, by displaying the name of the concerned household member and their label (inconsistency details through the property hshls:HasInconsistency).

VIII. RESULTS AND DISCUSSION

A. Results

Figure 6 illustrates the graph of actual data integrated, considering one household. We can see that the household member is linked to its household he belongs to, and has answers. This makes it possible to easily query information about a particular household, since the naming is simplified. In this illustration, we demonstrate how we can input the data of a household member in a way that provides ease of navigability through survey data.

Knowing the codification structure of household member identification information allows for straightforward formulation of data queries. In the case given in this figure (Figure 6), we have a household member whose name is “Annan” and sex “F” (for Female). That household member is identified by householdmember_11_516_4_2_3. We can easily navigate through that household member’s information using his identifier.

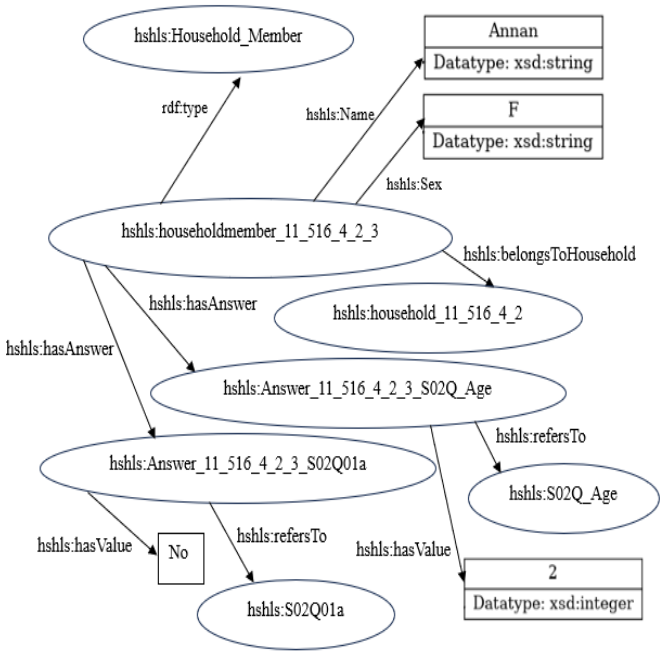


Figure 6: HSHLS graph visualization of actual data.

The following code gives an illustration of the result of the reasoning over the example of data given before. The result is saved in OWL in Turtle format (# denotes comments):

```
###
http://w3id.org/HshlsOnto/householdmem
ber_11_516_4_2_3
hshls:householdmember_11_516_4_2_3
rdf:type      owl:NamedIndividual      ,
hshls:Household_Member                 ,
hshls:Inconsistent                     ;
hshls:belongsToHousehold
hshls:household_11_516_4_2             ;
hshls:hasAnswer
hshls:answer_11_516_4_2_3_S02Q01a      ,
hshls:answer_11_516_4_2_3_S02Q_Age     ;
hshls:HasInconsistency    "Inconsistent
data. The age of this individual is
less than 3, therefore he or she must
not answer to questions on Education.
Please double check and correct the
information       accordingly."        ;
hshls:Name           "Annan"           ;
hshls:Sex "F" .
```

We can clearly see that this household member has been flagged as inconsistent, and this is in accordance with the rules we have defined. We now use the reasoning resulted graph to query information where there are inconsistencies.

For experimental purposes, we tested the performance of the reasoning engine on a small dataset of 60 household members (surveyed), described by 10 variables, including one quantitative variable (capturing the age of the surveyed) and nine qualitative variables. The data integration process took 0.5393 seconds.

Table V details the information (indicators and their respective values) sent to the reasoning engine as input.

TABLE V. STATISTICS OF INFORMATION SENT TO THE REASONING ENGINE

Indicator	Value
Number of SWRL rules exported to rule engine	9
Number of OWL class declarations exported to rule engine	9
Number of OWL individual declarations exported to rule engine	252
Number of OWL object property declarations exported to rule engine	6
Number of OWL data property declarations exported to rule engine	8
Total number of OWL axioms exported to rule engine	1321

The RDF input data model has 9 OWL class declarations, 252 OWL individual declarations, 6 OWL object property declarations, 8 OWL data property declarations, for a total number of 1321 OWL axioms. The transfer of all OWL axioms to the rule engine took 125 millisecond(s). The reasoning process took 517 millisecond(s).

The experimental dataset used contained 14 inconsistent values, and all inconsistencies were identified, resulting in a 100% success rate for the reasoner.

B. Discussion

This model makes it possible to store the methodological information of the Harmonized Survey on Households Living Standards in such a way that it can be understood by the computer and retrieved automatically. By specifying the semantics of the questions addressed, this model helps to better understand the meaning of the data manipulated in this survey as well as the semantic relationships that exist between these data. A data integration approach is also implemented. This allows inserting actual data in the model to make it possible to perform queries and reasoning on them. A reasoning process is proposed, through SWRL rules, that allows data consistency control during data collection and/or data processing. This serves as a fundamental tool for data quality control. Since the model is saved in a persistent repository, one can easily get access and perform some retrievals and analysis requests using SPARQL or any appropriate data analysis tool. Also, a large audience can get access and learn related knowledge. Therefore, the proposed solution will not only help improving the efficiency during the survey data collection and

processing activities, but also contributes to the dissemination of the survey knowledge.

This model highlights semantic information derived from the methodology of the Harmonized Survey on Households Living Standards. The reasoning system for data quality control is based on a set of SWRL rules. This allows detecting potential inconsistencies in actual data, in respect to the rules defined. However, since there are lot of rules for the entire survey, implementing all those rules would be a time-consuming task. Therefore, we propose to complement our system with a non-deterministic one: a machine learning system. Doing so, the rule-based system will not only be used for data inconsistency verification, but it will also involve in the validation process of the quality of the machine learning system to be built for the purpose of actual data quality control.

IX. CONCLUSION AND FUTURE WORK

In this work we propose a semantic data model of the Harmonized Survey on Households Living Standards, along with a way to integrate data and to query and reason over the built model. The model built makes it possible to store the semantic information contained in the methodology of this survey so that it can be consulted automatically. The results of this work can be exploited as part of the automatic retrieval of methodological information. Generally speaking, this work completes the state of the art and serves as a proof of concept to demonstrate the feasibility of documenting the knowledge contained in a statistical survey questionnaire through ontology-based semantic modeling. The work illustrates also the feasibility of data integration in an RDF-based model. The results of this work can also be used for data consistency control to improve the survey data quality. In the future, we will complement the built rule-based reasoning system with a machine learning approach, to discover hidden anomalies in actual data. As part of future work, we also propose the creation of a graphical user interface that will enable end-users to remotely query information from the model via SPARQL. This approach aims to provide an intuitive, user-friendly platform for exploring and querying ontological data without requiring users to manually interact with raw RDF files or directly write SPARQL queries.

For the model implementation, we will adopt an approach based on a knowledge base, enabling inference through an inference engine. This will be carried out using the Prolog programming language.

A program is a set of axioms, logical statements, that express the knowledge and hypotheses of the problem, and a calculation is a constructive proof of a goal based on the statements of the problem [21]. Ideally, the programmer expresses the logic of the problem to be solved, while the control is embedded or included in the interpreter of the language as such.

A logic program is a finite set of facts and rules, also called defined program clauses. A logic program is executed by assigning it a goal. Unification is the uniform mechanism for parameter passing, selection, and data construction.

Logic programming is based on the use of formal tools such as model theory and resolution theory to capture its semantics in a simple and efficient way [22]. Model theory is used to characterize the declarative semantics of the language, while resolution theory forms the basis of its operational semantics.

Prolog, as an implementation of logic programming is a powerful and simple language with a well-defined semantics, based on predicate calculus, offering several advantages such as unification and backtracking. Prolog language allows modeling knowledge using clauses (facts and rules) and inferring from this knowledge.

We will then compare the results of the two approaches, particularly in terms of performance and simplicity.

REFERENCES

[1] M. Mfoutou Moukala, M. Ngomo, and R. F. Babindamana, A Semantic Data Model of Harmonized Survey on Households Living Standards, SEMAPRO 2024 : The Eighteenth International Conference on Advances in Semantic Processing, 2024, pp. 1-7.

[2] Harmonized Survey on Households Living Standards 2018-2019, Food and Agriculture Organization of the United Nations, 2022, https://microdata.fao.org/index.php/catalog/2355/study-description (consulted on February 3, 2025).

[3] Semantic Web Rule Language. https://www.w3.org/submissions/SWRL/ (consulted on February 10, 2025).

[4] K. Munir and M. Sheraz Anjum, The use of ontologies for effective knowledge modelling and information retrieval, Applied Computing and Informatics, Vol. 14, N°2, pp. 116–126, 2018.

[5] R. Thirumahal, G. Sudha Sadasivam, and P. Shruti, Semantic Integration of Heterogeneous Data Sources Using Ontology Based Domain Knowledge Modeling for Early Detection of COVID-19, SN Computer Science, Vol. 3, No. 428, 2022, https://doi.org/10.1007/s42979-022-01298-4.

[6] I. Berges, J. Bermúdez, and A. Illarramendi, Towards Semantic Interoperability for Electronic Health Records, IEEE Trans Inf Technol Biomed, Vol. 16, N°3, pp. 424-431, 2012, doi: 10.1109/TITB.2011.2180917.

[7] N. C. Nicholson et al., An ontology-based approach for developing a harmonised data-validation tool for European cancer registration, Journal of Biomedicam semantics, Vol. 12, N°1, pp. 1-15, 2021.

[8] A. V. Borodin and Y. V. Zavyalova, An ontology-based semantic design of the survey questionnaires, 19th Conference of Open Innovations Association (FRUCT), 2016, Jyvaskyla, Finland, pp. 10-15.

[9] RDF 1.1 Primer, W3C Working Group Note 24 June 2014 https://www.w3.org/TR/rdf11-primer/, (consulted on February 3, 2025).

[10] RDF Schema 1.1, W3C Recommendation 25 February 2014, (Document updated on December, 1th 2023), https://www.w3.org/TR/rdf-schema/, (consulted on February 3, 2025)

[11] OWL2 Web Ontology Language, W3C Recommendation 11 December 2012, Document Overview (Second Edition),

https://www.w3.org/TR/owl2-overview/, (consulted on February 3, 2025).

[12] RDFLib. https://pypi.org/project/rdflib/, (consulted on February 3, 2025).

[13] PyLODE. https://pypi.org/project/pylode/, (consulted on February 10, 2025).

[14] M. Mfoutou Moukala, HSHLS survey ontology Web Page, https://moukmarc.github.io/.

[15] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz, Pellet: A Practical OWL-DL Reasoner, Journal of Web Semantics, Vol. 5, N°2, pp. 51-53, 2007.

[16] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang, HermiT: An OWL 2 Reasoner, Journal of Automated Reasoning, Vol. 53, pp. 245-269, 2014.

[17] D. Tsarkov and I. Horrocks, FaCT++ Description Logic Reasoner:System Description, 3rd International Joint conference on Automated Reasoning (IJCAR-2006), 2006, pp. 292-297.

[18] Protégé. https://protege.stanford.edu/software.php (consulted on February 3, 2025).

[19] Protégé 5 Documentation. http://protegeproject.github.io/protege/ (consulted on February 3, 2025).

[20] SPARQL 1.1 Query Language, W3C Recommendation 21 March 2013, https://www.w3.org/TR/sparql11-query/, (consulted on February 3, 2025).

[21] L. Sterling and E. Shapiro, L'Art de Prolog, Masson 1990.

[22] J. Jaffar, J.-L. Lassez, and M.J. Maher, A logic Programming Language Schemes, Logic Programming (DeGroot et Lindstrom), 1986, pp. 441-467.

APPENDIX A

Here is a part of the semantic model of the Harmonized Survey on Households Living Standards:

```
@prefix hshls: <http://w3id.org/HshlsOnto/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:<http://www.w3.org/2001/XMLSchema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix vann: <http://purl.org/vocab/vann/> .
hshls: a owl:Ontology;
   rdfs:seeAlso "https://github.com/moukmarc/HshlsOnto" ;
   dcterms:creator "Marc Mfoutou Moukala" ;
   dcterms:title "Harmonized survey on household living standards Ontology (HshlsOnto)" ;
   vann:preferredNamespacePrefix "hshls" .
# Core classes declaration
hshls:HSHLS a rdfs:Class ;
   rdfs:label " Harmonized Survey on Households Living Standards" .
hshls:Section rdf:type rdfs:Class ;
        rdfs:label " A section or module of the survey " .
hshls:Questionnaire rdf:type rdfs:Class .
hshls:Household a rdfs:Class ;
   rdfs:label " A household " .
hshls:Household_Member rdf:type rdfs:Class ;
   rdfs:label " A household member surveyed " .
hshls:Constraint rdf:type rdfs:Class ;
        rdfs:label " Constraint on the question " .
hshls:Question a rdfs:Class .
hshls:Answer a rdfs:Class .
# Properties declaration
hshls:Name a rdf:Property ;
   rdfs:domain hshls:Household_Member ;
   rdfs:range xsd:string .
hshls:Sex a rdf:Property ;
   rdfs:domain hshls:Household_Member ;
   rdfs:range xsd:string .
hshls:belongsToHousehold a rdf:Property ;
   rdfs:domain hshls:Household_Member ;
   rdfs:range hshls:Household .
hshls:clusternumber a rdf:Property ;
   rdfs:domain hshls:Household ;
   rdfs:range xsd:string .
hshls:departementNumber a rdf:Property ;
   rdfs:domain hshls:Household ;
   rdfs:range xsd:string .
hshls:hasAnswer a rdf:Property ;
   rdfs:domain hshls:Household_Member ;
   rdfs:range hshls:Answer .
hshls:hasConstraint a rdf:Property ;
   rdfs:domain hshls:Question ;
   rdfs:range hshls:Constraint .
hshls:hasQuestion a rdf:Property ;
   rdfs:domain hshls:Section ;
   rdfs:range hshls:Question .
hshls:hasSection a rdf:Property ;
   rdfs:domain hshls:HSHLS ;
   rdfs:range hshls:Section .
hshls:hasValue a rdf:Property ;
   rdfs:domain hshls:Answer ;
   rdfs:range rdfs:Literal .
hshls:householdNumber a rdf:Property ;
   rdfs:domain hshls:Household ;
   rdfs:range xsd:string .
hshls:refersTo a rdf:Property ;
   rdfs:domain hshls:Answer ;
   rdfs:range hshls:Question .
hshls:waveNumber a rdf:Property ;
   rdfs:domain hshls:Household ;
   rdfs:range xsd:string .
# Model specialization
hshls:HSHLS-C1 rdf:type hshls:HSHLS ;
        hshls:hasSection hshls:S00, hshls:S01, hshls:S02, hshls:S03, hshls:S04, hshls:S05, hshls:S06, hshls:S07,
        hshls:S08, hshls:S09, hshls:S10, hshls:S11, hshls:S12, hshls:S13, hshls:S14, hshls:S15, hshls:S16, hshls:S17,
        hshls:S18, hshls:S19, hshls:S20, hshls:S21 ;
        rdfs:comment "The different sections of HSHLS survey for the first edition in Congo named here HSHLS-C1" .
# Questions in section S02
hshls:S02 a hshls:Section ;
hshls:hasQuestion hshls:S02Q_Age, hshls:S02Q01a, hshls:S02Q01b, hshls:S02Q01c, hshls:S02Q01d,
```

```
hshls:S02Q02a, hshls:S02Q02b, hshls:S02Q02c,
hshls:S02Q02d, hshls:S02Q03a, hshls:S02Q03b,
hshls:S02Q03c, hshls:S02Q03d, hshls:S02Q04,
hshls:S02Q05, hshls:S02Q06, hshls:S02Q07,
hshls:S02Q08, hshls:S02Q09, hshls:S02Q10,
hshls:S02Q11, hshls:S02Q12, hshls:S02Q13,
hshls:S02Q14, hshls:S02Q15, hshls:S02Q16,
hshls:S02Q17, hshls:S02Q18, hshls:S02Q19,
hshls:S02Q20, hshls:S02Q21 ;
rdfs:comment "This section captures household member's education information for members of 3 years old and more" .
# Constraint for valid responses for the question S02Q01a
hshls:ConstraintS02Q01a a hshls:Constraint ;
hshls:validValues "Yes","No" .
# Constraint for valid responses for the question S02Q03
hshls:ConstraintS02Q03 a hshls:Constraint ;
hshls:validValues "Yes","No, never attended" .
# Example of survey data integrated
hshls:householdmember_11_320_2_2_1 a
hshls:Household_Member ;
   hshls:Name "Lotté"^^xsd:string ;
   hshls:Sex "M"^^xsd:string ;
   hshls:belongsToHousehold
hshls:household_11_320_2_2 ;
   hshls:hasAnswer
hshls:answer_11_320_2_2_1_S02Q01a,
      hshls:answer_11_320_2_2_1_S02Q03,
      hshls:answer_11_320_2_2_1_S02Q_Age .
hshls:householdmember_11_324_4_2_2 a
hshls:Household_Member ;
   hshls:Name "Bruno"^^xsd:string ;
   hshls:Sex "M"^^xsd:string ;
   hshls:belongsToHousehold
hshls:household_11_324_4_2 ;
   hshls:hasAnswer
hshls:answer_11_324_4_2_2_S02Q01a,
      hshls:answer_11_324_4_2_2_S02Q03,
      hshls:answer_11_324_4_2_2_S02Q_Age .
hshls:householdmember_11_516_4_2_1 a
hshls:Household_Member ;
   hshls:Name "Mouk"^^xsd:string ;
   hshls:Sex "M"^^xsd:string ;
   hshls:belongsToHousehold
hshls:household_11_516_4_2 ;
   hshls:hasAnswer
hshls:answer_11_516_4_2_1_S02Q01a,
      hshls:answer_11_516_4_2_1_S02Q_Age .
hshls:householdmember_11_516_4_2_2 a
hshls:Household_Member ;
   hshls:Name "Marc"^^xsd:string ;
   hshls:Sex "F"^^xsd:string ;
   hshls:belongsToHousehold
hshls:household_11_516_4_2 ;
   hshls:hasAnswer
hshls:answer_11_516_4_2_2_S02Q01a,
      hshls:answer_11_516_4_2_2_S02Q_Age .
hshls:householdmember_11_516_4_2_3 a
hshls:Household_Member ;
   hshls:Name "Annan"^^xsd:string ;
   hshls:Sex "F"^^xsd:string ;
   hshls:belongsToHousehold
hshls:household_11_516_4_2 ;
   hshls:hasAnswer
hshls:answer_11_516_4_2_3_S02Q01a,
      hshls:answer_11_516_4_2_3_S02Q_Age .
hshls:refersTo a rdf:Property ;
   rdfs:domain hshls:Answer ;
   rdfs:range hshls:Question .
hshls:waveNumber a rdf:Property ;
   rdfs:domain hshls:Household ;
   rdfs:range xsd:string .
hshls:answer_11_320_2_2_1_S02Q01a a hshls:Answer ;
   hshls:hasValue "Yes" ;
   hshls:refersTo hshls:S02Q01a .
hshls:answer_11_320_2_2_1_S02Q03 a hshls:Answer ;
   hshls:hasValue "Yes" ;
   hshls:refersTo hshls:S02Q03 .
hshls:answer_11_320_2_2_1_S02Q_Age a hshls:Answer ;
   hshls:hasValue 56 ;
   hshls:refersTo hshls:S02Q_Age .
hshls:answer_11_324_4_2_2_S02Q01a a hshls:Answer ;
   hshls:hasValue "No" ;
   hshls:refersTo hshls:S02Q01a .
hshls:answer_11_324_4_2_2_S02Q03 a hshls:Answer ;
   hshls:hasValue "No, never attended" ;
   hshls:refersTo hshls:S02Q03 .
hshls:answer_11_324_4_2_2_S02Q_Age a hshls:Answer ;
   hshls:hasValue 8 ;
   hshls:refersTo hshls:S02Q_Age .
hshls:answer_11_516_4_2_1_S02Q01a a hshls:Answer ;
   hshls:hasValue "No" ;
   hshls:refersTo hshls:S02Q01a .
hshls:answer_11_516_4_2_1_S02Q_Age a hshls:Answer ;
   hshls:hasValue 51 ;
   hshls:refersTo hshls:S02Q_Age .
hshls:answer_11_516_4_2_2_S02Q01a a hshls:Answer ;
   hshls:hasValue "No" ;
   hshls:refersTo hshls:S02Q01a .
hshls:answer_11_516_4_2_2_S02Q_Age a hshls:Answer ;
   hshls:hasValue 32 ;
   hshls:refersTo hshls:S02Q_Age .
hshls:answer_11_516_4_2_3_S02Q01a a hshls:Answer ;
   hshls:hasValue "No" ;
   hshls:refersTo hshls:S02Q01a .
hshls:answer_11_516_4_2_3_S02Q_Age a hshls:Answer ;
   hshls:hasValue 2 ;
   hshls:refersTo hshls:S02Q_Age .
hshls:household_11_320_2_2 a hshls:Household ;
   hshls:clusternumber "320"^^xsd:string ;
   hshls:departementNumber "11"^^xsd:string ;
   hshls:householdNumber "2"^^xsd:string ;
   hshls:waveNumber "2"^^xsd:string .
hshls:household_11_324_4_2 a hshls:Household ;
   hshls:clusternumber "324"^^xsd:string ;
   hshls:departementNumber "11"^^xsd:string ;
   hshls:householdNumber "4"^^xsd:string ;
   hshls:waveNumber "2"^^xsd:string .
```

```
hshls:household_11_516_4_2 a hshls:Household ;
   hshls:clusternumber "516"^^xsd:string ;
   hshls:departementNumber "11"^^xsd:string ;
   hshls:householdNumber "4"^^xsd:string ;
   hshls:waveNumber "2"^^xsd:string .
```

APPENDIX B

Here are some Semantic Web Rule Language rules implemented over the semantic model of the Harmonized Survey on Households Living Standards:

```
hshls:Household_Member(?HMember) ^ hshls:hasAnswer(?HMember, ?answer) ^ hshls:refersTo(?answer, hshls:S02Q_Age) ^ hshls:hasValue(?answer, ?value) ^ swrlb:lessThan(?value, 3) ^ hshls:hasAnswer(?HMember, ?answer2) ^ hshls:refersTo(?answer2, hshls:S02Q01a) -> hshls:Inconsistent(?HMember) ^ hshls:HasInconsistency(?HMember, "Inconsistent data. The age of this individual is less than 3, therefore he or she must not answer to questions on Education. Please double check and correct the information accordingly.")

hshls:Household_Member(?HMember) ^ hshls:hasAnswer(?HMember, ?answer) ^ hshls:refersTo(?answer, hshls:S02Q_Age) ^ hshls:hasValue(?answer, ?value) ^ swrlb:lessThan(?value, 3) ^ hshls:hasAnswer(?HMember, ?answer2) ^ hshls:refersTo(?answer2, hshls:S02Q01b) -> hshls:Inconsistent(?HMember) ^ hshls:HasInconsistency(?HMember, "Inconsistent data. The age of this individual is less than 3, therefore he or she must not answer to questions on Education. Please double check and correct the information accordingly.")

hshls:Household_Member(?HMember) ^ hshls:hasAnswer(?HMember, ?answer) ^ hshls:refersTo(?answer, hshls:S02Q_Age) ^ hshls:hasValue(?answer, ?value) ^ swrlb:lessThan(?value, 3) ^ hshls:hasAnswer(?HMember, ?answer2) ^ hshls:refersTo(?answer2, hshls:S02Q01c) -> hshls:Inconsistent(?HMember) ^ hshls:HasInconsistency(?HMember, "Inconsistent data. The age of this individual is less than 3, therefore he or she must not answer to questions on Education. Please double check and correct the information accordingly.")

hshls:Household_Member(?HMember) ^ hshls:hasAnswer(?HMember, ?answer) ^ hshls:refersTo(?answer, hshls:S02Q_Age) ^ hshls:hasValue(?answer, ?value) ^ swrlb:lessThan(?value, 9) ^ hshls:hasAnswer(?HMember, ?answer2) ^ hshls:refersTo(?answer2, hshls:S02Q01a) -> hshls:Inconsistent(?HMember) ^ hshls:HasInconsistency(?HMember, "Inconsistent data. The age of this individual is less than 9, therefore he or she must not answer to questions on Languages. Please double check and correct the information accordingly.")
```

Containerization's Power Use Overhead in Video Streaming — and the Case for Optimizing Scheduler Power in Tickless Kernels

Etienne-Victor Depasquale
Department of Communications and Computer Engineering
University of Malta
Msida, Malta
e-mail: edepa@ieee.org

Saviour Zammit
Department of Communications and Computer Engineering
University of Malta
Msida, Malta
e-mail: saviour.zammit@um.edu.mt

Abstract— **Containerization of a service enables live migration and, thereby, consolidation of running service instances onto as few host platforms as possible. However, containerization's operational overhead must be investigated to determine overall viability. One dimension of this overhead is that of power use, and this is investigated here. An architecture for a video cache service at the edge of a Communications Service Provider's (CSP) network in the metropolitan area is designed, and a scaled version is implemented in a laboratory environment. A comparison is made between power used while streaming videos in both native and containerized modes of operation. Containerization is found to incur a low power overhead while streaming video, compared with streaming video from ffmpeg running directly on the host operating system. Moreover, notwithstanding advances with tickless kernels, the kernel's scheduling timer routine remains a dominant power consumer. Power use is measured using hardware instrumentation and with PowerTOP, a software power meter. Limits on the latter's accuracy have been observed.**

Keywords- containers; power; video; streaming; implementation model; tickless kernel; PowerTOP; power instrumentation.

I. INTRODUCTION

This paper expands upon our earlier study presented at the Ninth International Conference on Green Communications, Computing and Technologies (GREEN 2024) [1], where we investigated the power overheads incurred by containerized video streaming.

Content Delivery Networks (CDNs) are overlay networks that are key to controlling the growth in demand for bandwidth in long-haul communications links. By distributing content to caches in geographical regions of the world where customers are located, the number of times which a single item of content crosses long-haul links between the content origin's region and the customer's region, is reduced to just one. In turn, the content is distributed several times to customers in the region. While the function of the CDN, from the customer's perspective, is that of reducing latency and avoiding buffer underrun, the control of bandwidth growth is a function that has a strategic role in the stability of world-wide communication. The CDN's role in bandwidth control continues to gain attention [2]; a variety of CDN implementations has been investigated [3], [4] and surveyed [5], [6] and generalized surveys are of ongoing interest [7], [8]. The importance of the CDN seems to grant sufficient ground for study of the impact of its Point of Presence (PoP) on the information and communication technology of its environs.

This study seeks to compare power use in containerized deployment of the media server in a CDN PoP. It focuses on the power use of the media server as it processes a representative set of tasks. The media selected for study is video (henceforth, the media server will be referred to as the video server), and two reasons support this choice. Video dominates traffic, whether in the access, aggregation, metro-core, or long-haul. Moreover, some of the tasks, such as transcoding, are processor-intensive and serve to indicate the power capacity required to support CDN PoPs.

The rest of this paper is structured as follows. Section II is a brief treatise of some of the foundational works in modelling power use in computer systems. In Section III, the objective is stated. In Section IV, the implementation model is presented. This supports reproduction of the test environment. In Section V, the method is elaborated upon. Section VI presents the results, and Section VII supports interpretation through analysis of these results. Section VIII draws a succinct conclusion on the impact that containerization of a video service has on power use overhead.

II. BACKGROUND

A. Power models

A better grasp of the impact of containerization on a CDN PoP's power use requires understanding of containerization's impact on the media server's use of power. Media servers are deployed on general purpose computing systems (or: Commercial-Off-The-Shelf (COTS) computer systems), and a basic understanding of these systems' power characteristics must support further study.

Power used by a computer system has an idle (static, or leakage) part, an active (dynamic) part and overhead. The power use referred to here is a system metric: it is an aggregate that sums all consumers' (system components) power use, whether it be of dynamic, static or overhead type. Idle power's (P_{idle}) relevance depends on perspective. On the one hand, idle power use is a real and significant cost: from the perspective of facility managers and sustainability advocates, it is of interest. On the other hand, it is irrelevant to the way in which processes exercise a computer system: it is of

secondary importance in a study of the power required to deliver a service. It follows, then, that a study of the power used to operate a CDN PoP's server systems must primarily engage with service power use, or, using general terms, with dynamic power use.

A simple, yet useful, classification of power models divides them into two: (a) one that treats the computer system as a black box, and uses workload to predict power in real time, and (b) another that exploits knowledge of microarchitecture and architecture [9]. They represent two different levels of abstraction (see [10, p. 42] for a more detailed distinction) of a computer system.

1) The affine relationship between aggregate power and utilization

The affine relationship is a well-known example of the black-box class. A typical representation of workload may be one or more parameters of utilization (e.g., MIPS (millions of instructions per second) and IOPS (input/output operations per second). The affine relationship between power use and utilization [9] is well suited to describing legacy network equipment [11]. The general form is reproduced as equation (1):

$$P(\rho) = P_{idle} + \big(P(\rho = 1) - P(\rho = 0)\big)\,\rho \qquad (1)$$

where P(ρ) expresses power at utilization ρ.

Equation (1) expresses power in terms of generalized utilization, but particular forms like processor load in MIPS or switching throughput in bits per second may be used (where the model holds true). Note that equation (1) refers to the idle power (P_{idle}), but not to the overhead. The static part and overhead are significant and cannot be ignored. However, idle power use has no correlation with the computer system's load. Furthermore, while the overhead (such as fan power use) can indeed be expected to relate to load (it is not a constant type of overhead), power used by these overheads can be expected to have much longer response times than that of power used by silicon. For example, a fan's speed will increase when the temperature in a thermally instrumented zone increases; heat capacity is clearly a factor that will affect temperature rise, as well as temperature drop. Therefore, fan speed will not follow silicon loading and inclusion of fans' power use in measurements will obfuscate the dynamics of power use by silicon under load.

2) Limitations of the affine relationship

Accuracy of the affine relationship has been shown to worsen when the processor does not dominate dynamic power [9]. Apart from processor utilization, system power models have been developed to handle other system components like primary (silicon dynamic random-access memory) and secondary storage (disks) [12]. Furthermore, processor power and frequency are quadratically related [13]. One approach to handling frequency variability is given in [14], where the affine relationship is modified and takes the form shown in equation (2):

$$P(\rho) = P_{idle}^{(f)} + \Big(P^{(f)}(\rho = 1) - P^{(f)}(\rho = 0)\Big)\,\rho \qquad (2)$$

In equation (2), the frequency index in $P_{idle}^{(f)}$ and $\Big(P^{(f)}(\rho = 1) - P^{(f)}(\rho = 0)\Big)$ serves to denote the dependence of intercepts (static/leakage/idle power) and gradients on frequency of operation. Evidently, the affine model expressed in equation (1) does not describe a computer system's processor's power use when the processor is operating under dynamic adaptation of voltage and frequency (Dynamic Voltage and Frequency Scaling (DVFS)). However, system power measurements must be adjusted by a baseline that includes $P_{idle}^{(f)}$. A corresponding measure will be dealt with in considerations of measurement methodology.

3) Relevance of the architectural models

Power architectural models predict power use as a linear function of several activity indicators. These indicators regard the activity of aspects of architecture and microarchitecture of a system. Therefore, models that harness microarchitectural activity indicators tend to be bound to specific hardware models. Activity indicators are commonly referred to as performance counters. Architectural models are well suited to the task of measuring dynamic power use. Performance counters compiled by the operating system are architectural; those compiled by the hardware in dedicated registers, are microarchitectural. System software can abstract microarchitectural counters by a layer that returns these counters through method calls.

A particularly useful class of these counters obtains power use directly. Examples include Intel's Running Average Power Limit (RAPL) and AMD's AMD Energy Driver (amd_energy). The feature addresses power use through hardware support and can form part of a measurement methodology. For example, RAPL's MSR_PKG_ENERGY_STATUS register provides a running, cyclic total of energy used by the CPU (Central Processing Unit) package (all cores included).

B. Isolation and attribution of dynamic power use

Dynamic power is used during both service idle time and service delivery time. The power used during service idle time is not the P_{idle} in equation (1). Rather, it is the dynamic power used by system software (whether in user or kernel mode) to maintain system operation. Service idle time's power use will be subtracted from service delivery (during video streaming) time's power use, to obtain the differential relevant to this research. Service idle time's power use is a tangible justification of the requirement to use minimalist general-purpose systems. Since user application and system software processes and threads are many, then minimization of such root causes simplifies the process of attribution of dynamic power use. An illustration of the multiplicity of subsystems of the computer that are in the scope of power use measurement may be found in [15]; clearly, detailed inspection is required to correctly isolate and attribute power use. Two broad classes of process and thread are identifiable and are briefly described below. Sub-sections II-B-1 and II-B-2 concern the video server, but the same considerations readily hold for the virtual switch's server too.

1) Kernel operations

There are several categories of operation carried out by the kernel to support the operations of the video service. These include:

- processing of hardware interrupts when packets arrive, and concomitant activities like the onerous requirement for the system call to return to userspace;
- managing the timer, to schedule processor allocation to processes and threads;
- memory and cache management, and
- processing of system power monitoring instructions.

The detail of which processes to monitor is expected to be captured in the baseline (see approach-baselining, below). The power used during service idle time will be subtracted from power used during service delivery (during video streaming), to obtain the differential relevant to this research.

2) Service operations

The video streaming service may be tersely described as one in which:

- a source file is encoded (or transcoded) using a video codec and an audio codec;
- the codecs' output is packetized and
- transmitted over a network interface as the payload of a communication protocol that handles:
    - the correct sequencing of the received content (payload) and
    - adaptation of the video quality of the content to network conditions.

These operations must be matched to specific computing entities (components such as processes, threads, main memory and cache) in the computer system and the power use thereof is to be monitored. In particular, the specific computing entities are expected to include the video server process obtained by running the principal executable, and library functions which it calls to support the three major categories of operation listed above (encoding, packetization and sequencing into a stream of adaptable quality).

C. Service scaling

Service scalability is essential to cope cost- and energy-efficiently with short-term fluctuations in demand. These fluctuations are commonly referred to as the daily diurnal and nocturnal crests and troughs In Internet service demand. Figure 1 [16] illustrates service scaling in a virtualization infrastructure. The range of service supply varies from minimum scaleLevel to maximum scaleLevel, stepping with the size corresponding to a virtual network function component (VNFC). Higher demand (load) can be met by spawning one service instance per client. The service instance may consist solely of a single VNFC.

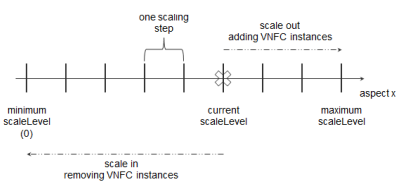


Figure 1. [16, Fig. 5.1–1] Demand is met by deploying over a range from minimum to maximum scaleLevel

III. OBJECTIVE

A. Principle

An overhead is expected in the containerized implementation, and its quantification is sought. The objective can be articulated in terms of a comparison between two types of deployment:

- power use in a computer system that runs the service within containers, with
- power use in a computer system that runs the service directly on the operating system.

Quantification is sought to control a tradeoff between native and containerized deployment. The tradeoff may be succinctly summarized as one of greater operating power per unit (physical host) versus potential for lower number of operating units (physical hosts). The following sub-sections elaborate on this summary.

B. Greater operating power per unit

1) Quiescent operating power

A host (physical server) computer system uses power in its quiescent state. Quiescence is the condition where the host has an active operating system and is running a minimal set of services. Levels of quiescence can be defined, in accordance with different specifications of the set of services. In all levels, quiescent power use consists of an idle/leakage/static component, due solely to physical properties of the hardware, and a dynamic component, due to execution of software processes on the hardware. A containerized deployment uses more power in the quiescent state because its minimal set of services is a superset of that used by a native deployment. Therefore, even when no video clients are served, a containerized deployment has greater operating power. Moreover, to grasp the difference between operating power of the two deployments, a service process deployment strategy must be defined.

2) Full load operating power

As clients appear, service processes must be started to handle the workflow. Minimally, the video service workflow consists of the following cyclical process:

- fill a memory buffer queue by copying some initial large chunk of the file from storage;
- transmit the queue head;
- repeat queue head transmission until some fraction of the queue is empty;
- re-fill the memory buffer queue from memory ramdisk and
- repeat the second, third and fourth steps until all of the file has been read into the queue.

The process, which can be tersely summarized as streaming, is independent of the file's encoding format, but will be extended should real-time transcoding be necessary to meet the client's constraints. These observations prompt the identification of load units, comprising the full amount of work (in Joules) required to process the workflow. A topmost classification divides the set of load units into two branches:

one for the case where only streaming is needed, and another for the case where both streaming and real-time transcoding is needed. Below this topmost classification, load units can be identified for every encoding type and bitrate preset. Each such load unit corresponds to a single resource unit, which is the bundle of computing and networking resources required to serve the load unit. Specification of a load unit supports the analysis of full load operating power, as the latter is used when no more load units can be taken (subject to some quality of service (QoS) condition, as described below). This limit can be articulated better in terms of bin packing, where each physical host is represented as a bin capable of serving load. When a load unit is served, the bin is partially filled, and the corresponding resource unit is removed from the aggregate of the host's available resources. As more load units are served, the bin is progressively filled until no more load units can be added. This is full load, and the power used under this condition is the full load operating power.

3) Bin packing

The condition of full load corresponds to the operating principle of maximization of capacity utilization without degrading key indicators of quality of service (QoS). That is, if each server represents a bin of some service capacity C, then the server is loaded until its capacity is fully utilized without degrading the QoS. The process of filling the server suggests modelling using bin-packing algorithms; hence, depiction of the server as a bin.

Since both capacity, C, and QoS are complex, a simplification is sought to manage the tractability of the problem. Let the capacity, C, be the number L of load units U that a host H in a set of homogenous hosts, can serve without degrading the received bit rate at any client, below the preset for U. This specification of C and QoS key performance indicator (KPI) serves to support specification of other, different loading conditions for both types of deployment. This is reserved for future study.

C. Potential for lower number of operating units

Consider the condition of consolidation, obtained by deploying video service process to the minimum number of hosts possible. Ideal consolidation is obtained when all N hosts (bins) in service except for the Nth are packed. Here, bin-packing corresponds to loading a server until the bit rate served at one or more of the clients falls below the preset.

Such an idealized consolidation is depicted in Figure 2. The top part (a) shows ideal consolidation, at some time t = 0. $N(@t = 0)$ (henceforth denoted by $N(0)$) servers are shown, of which $N(0) - 1$ are filled and 1 is partially filled. One white segment represents one utilized resource unit. One context in which this consolidation is achievable is when an initial set of load units is presented to a dispatching subsystem for distribution onto a set of idle servers. This context applies to both the case where the service is running as a User Application (UA) directly on the host Operating System (OS) (henceforth shortened to "running as a UA") and the case where the service is running in a container.

Over time, clients drop out (the black gaps represent unutilized resource units) as their viewing sessions end. While running as a UA, the service instance supporting dropped cl-

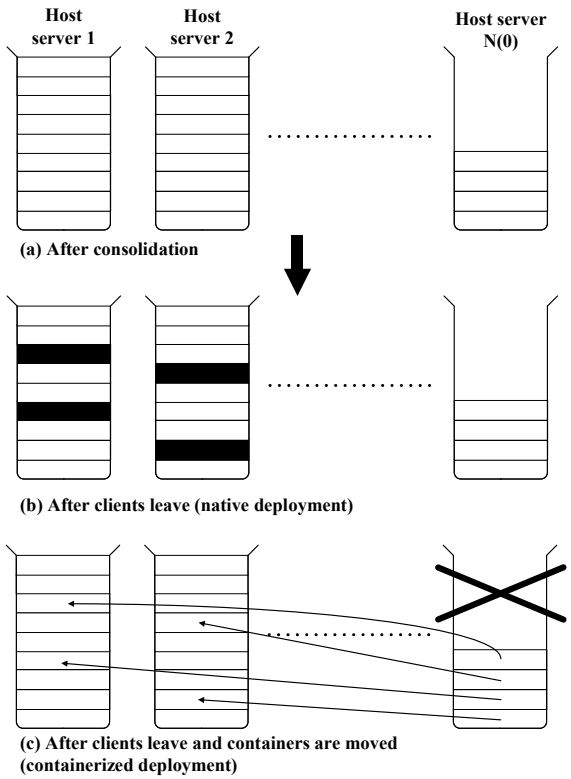


Figure 2. Simplified view of power control enabled by containerization of service application

-ients terminates and leaves a resource gap. However, these gaps cannot be filled with running instances on other servers, since UA state cannot be migrated as easily as when it runs within a container. At some arbitrary time, t, after service starts, it may not be possible to consolidate the service running as a UA, but it should always be possible to consolidate the service running containerized. Therefore:

- while server $k \in \{1, 2, \ldots N(0)\}$, draws $P_k^{(c)} > P_k^{(ua)}$, where $P_k^{(c)}$ represents power drawn while serving a capacity-sized subset from containers and $P_k^{(ua)}$ is the native counterpart, and
- while $N^{(c)}(0) \geq N^{(ua)}(0)$, since $L^{(c)} \leq L^{(ua)}$, where $L^{(c)}, L^{(ua)}$ are the respective capacities of the containerized and native service deployments,

there is no predetermined relationship between $N^{(c)}(t)$ and $N^{(ua)}(t)$. Moreover, while all but the last of the $N^{(c)}(t)$ hosts can (at least periodically) be subjected to consolidation and thus use power amounting to $P_k^{(c)}$ W, the operating state of the $N^{(ua)}(t)$ hosts, and their individual power uses, is unknown. It thus follows that the relationship between $P_{N^{(c)}(t)}^{(c)} + \left(N^{(c)}(t) - 1\right)P_k^{(c)}$ and $\sum_{k=0}^{N^{(ua)}(t)-1} P_k^{(ua)}$ is not evident, and **quantification** of the overhead $P_k^{(c)} - P_k^{(ua)}$ due to containerization, is a necessary prerequisite to understanding the scale and usage pattern at which

containerized deployment is energy efficient compared with native deployment.

IV. IMPLEMENTATION MODEL

An edge cache of a video streaming service is deployed. A high-level view of the implementational model is shown in Figures 3 and 4.

- Figure 3 shows an implementation that is easily portable to a cloud-native infrastructure (henceforth referred to as the cloud-native implementation), and
- Figure 4 shows an implementation that is a hybrid of physical (the video server) and virtual network functions (the switch).

The cloud-native implementation uses containers to host the video server. Both implementations host a virtual layer 2 switch in the intermediate node.

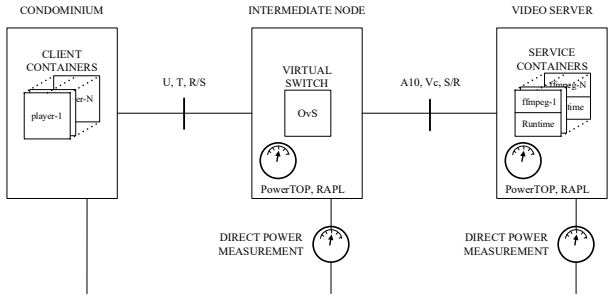


Figure 3. Physical topology of the video streaming service, deployed in containers. Video Server located in local exchange or Access Node (AN); Intermediate Note located in street cabinet (subtended AN [17]).

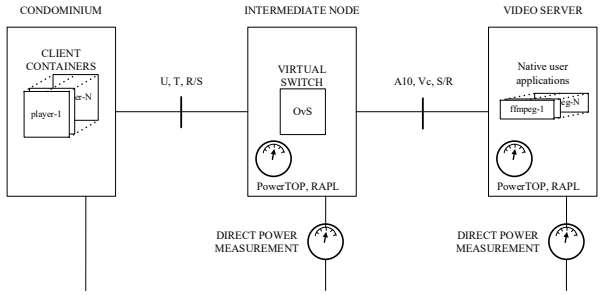


Figure 4. Physical topology of the video streaming service, deployed on a host operating system.

A. Hardware

The hardware used in this testbed consists of a set of three HPE (Hewlett Packard Enterprise) ProLiant BL460c Gen9 blade servers [18], hosted in an HPE c7000 blade enclosure. Connectivity between server and client blades is obtained through pass-through interconnect bay modules, patched with single-mode optic fibre cables. These latter modules support the goal of bypassing c7000 ecosystem interconnect-bay physical networking devices. Bypass is necessary to introduce separate, **virtual** switching hardware. The virtual switch is implemented on a third HPE Gen9 blade server. The links to the switch are of type 10GBASE-SR. The video server has a single Intel® Xeon® CPU E5-2640 v3 (2.60GHz) processor package. Dynamic Voltage and Frequency Scaling (DVFS) is under system firmware control.

B. Software

The software consists of:

- an FFmpeg [19]video server. This is representative of the access node at the edge of the metro-core network;
- a TSDuck [20] receiver. This is representative of end-user's video player, and is also used to measure received bitrate to ensure that Quality of Service (QoS) (see Section IV-C) is respected;
- the virtual switch software is Open vSwitch [21].

A minimalist operating system was selected for the video server, to support isolation and attribution in power measurements. While minimalist operating systems do not necessarily correlate with minimal noise in power measurement, it seems useful to reduce the number of possible sources from the outset. For this reason, Alpine Linux [22] Standard distribution version 3.19 was chosen.

The container system software selected is Docker [23]. Docker is a mature containerization platform and it is modular: the runtime daemon (containerd) supports other user interfaces apart from the Docker user interface (dockerd). For example, Kubernetes [24] can be used to manage containers created through the Docker Command-Line Interface (CLI).

V. METHOD

A. Instrumentation

Near-real time measurement of power use can be obtained from two sources of instrumentation. The blade servers are equipped with a management processor (known as "integrated lights-out", or iLO) that logs a power measurement every 10 seconds and stores a 20-minute history that can be read through a Redfish®[25] - compliant RESTful (Representational State Transfer) Application Programming Interface (API). Selectivity in aggregate power use measurement is afforded by blade systems, since these separate power supply to the (blade) computer system from power supply to two major overhead power drains. Blade servers use blade chassis services for power supply (where ac – dc conversion losses occur) and cooling (where blowers use power as they ventilate from chassis front to chassis rear). Thus, measurement of power used by the blade server at the supply voltage rails is free of the problematic, variable contribution from overheads, and idle power can be measured to the accuracy afforded by these blade system power measurement instruments. The measurement datum is of integer type, obtained by truncation of the decimal part of the actual measurement. Moreover: since the iLO is not part of the System Under Test (SUT), it does not alter power measurement.

While the iLO provides an aggregate power measurement, process- and thread- level granularity is obtained through

software power meters. Hardware extensions for power measurement are available in processor models that support the Intel Running Average Power Limit (RAPL) feature. PowerTOP [26] is software that enables this level of power attribution, and it is indeed capable of exploiting RAPL. This tool complements the aggregate power measurement obtained by blade sensor instrumentation. PowerTOP uses a top-down approach [27], (it divides the power measurement over a period amongst processes and threads in proportion to their core utilization) and precedes the measurement period by one of calibration (the utility was run in calibration mode for several hours before starting the first experiment) in which it obtains weighting parameters for the attribution process. Calibration is further refined with use, and PowerTOP saves its parametric refinement to persistent storage for future exploitation [28]. PowerTOP was used in its logging mode of operation, with 10 (ten) – second averaging intervals. However, PowerTOP has several significant limitations, as follows: it only measures dynamic power, it does not capture all power use, and it increases the SUT's aggregate power use. These must be mitigated.

B. Baselining

It is necessary to distinguish power used by the video service from power used by other consumers. This requires measurement of static (/idle) power use. It is also necessary to distinguish between dynamic power used during video service operation time, from dynamic power used when the service is idle. In essence: service power use can be thought of as an amount added above that used by the operating system and system software, which in turn is added above that used to operate electronic components (static/idle) power. Hence, it is possible to perceive a baseline to which service power is added to obtain the total power. Formally:

$$P_{b_1}^{(video)} = P_{idle}^{f_1} + P_q^{(os)}$$

where $P_q^{(os)}$ is the **dynamic** power corresponding to the OS's operation **without** container system software and without running User Applications (UAs), and $P_{idle}^{f_1}$ is the **idle/static** power at the frequency f_1 at which the OS is quiescent.

A second baseline, $P_{b_2}^{(video)}$, is required to ensure experimental reproducibility: it defines known starting and ending points for each run of experimentation.

$$P_{b_2}^{(video)} = P_{idle}^{f_2} + P_q^{(os+dockerd+containerd)}$$

Here, $P_q^{(os+dockerd+containerd)}$ is the **dynamic** power corresponding to the OS's operation **with** container system software but without running User Applications (UAs), and $P_{idle}^{f_2}$ is the **idle/static** power at the frequency f_2 at which the Operating System (OS) is quiescent. The state of quiescence is defined below (see V-D-2).

C. Mitigating errors

The principal source of error is measurement uncertainty at the iLO, as the iLO rounds to the nearest integer. Since the iLO rounds $[n-0.5, n+0.5)$ to $n, \in \mathbb{N}$, then, without further information on the probability density function (pdf) of the error, a fair representation of each measurement is the value n obtained by the iLO. This contrasts with the floor (round down/truncation) function, where a fair representation of a measurement n would be $n+0.5$, or the ceil (round up) function, where $n-0.5$ would be fair. Of the three conversions from real to natural number representation, rounding to the nearest integer has the least maximum error, and this corresponds to 0.5 W.

The ideal statistical distribution of errors is that of a uniform Probability Density Function (PDF). If measurement errors were indeed so distributed, then the mean of actual measurements can be obtained as the mean of the set of errored measurements. However, for the specific operating context of a quiescent operating system, the probability of a non-uniform distribution cannot be neglected because the dynamic power is low enough to keep the total power's range within half a watt. This is prone to persistent positive bias in error or persistent negative bias. In such non-uniform PDFs, the actual mean cannot be obtained; only a range of values within which the actual mean lies, can be obtained.

Both baselines regard quiescent states. If bias is detected, mitigation can be pursued through the less biased of the two baselines. The better baseline can be used to compute the affected baseline as the arithmetic combination (addition/subtraction) of the better baseline and the difference in dynamic power between the two baselines. Therefore, each measurement of baseline power must be accompanied by a measure of dynamic power, to support evaluation of the error in the means obtained through the iLO's measurements.

This approach notwithstanding, it may still not be possible to reconcile the two baselines in this manner. In such an eventuality, the ranges of values within which the actual means lie can be combined with the difference in dynamic power between the two baselines. The objective remains that of reconciling all measurements, within the margin of error anticipated.

D. Quality of Service

QoS is considered to be satisfied as long as there is sufficient capacity in the links to keep the overall average received bitrate of every video stream at or above the video file's overall bitrate.

E. Experiments

1) Test conditions

Video service will be delivered from both containerized and native deployments. The test conditions pertinent to the video server will be the following.

1. Implementation
   a. During containerized operation, each video service process and the libraries on which it depends will be operated from a container. One service process serves one client.
   b. During native operation, a new instance of the video service process will be started for every new client.
2. Load unit: This will consist of the work required to process a workflow based upon a video with the following technical specifications:

a. Overall bitrate = 457 kb/s, = video bitrate of 326 kb/s + audio bitrate of 127 kb/s + mp4 container metadata rate (overhead)
b. Duration = 1h 32m 2.19s (5522.19 s), of which 30 minutes are played, starting at a randomly selected point in the video.
c. H.264 video codec, Main profile
   i. Resolution = 1280 x 720
   ii. Frame rate ≈ 23.98 frames/second (fps)
d. Advanced Audio Coding (AAC) audio codec, Low Complexity profile
   i. Sampling rate = 44.1 kHz
e. Client supports same video and audio codec; hence server does not need real-time transcoding.

2) Procedure

The power used by the video server is measured at progressively higher load levels. Two sets of experiments are carried out: the first set uses containerized video server instances and the second set uses native video server instances. A containerized service instance consists of a container carrying ffmpeg. A single container is created to deliver a single stream and is destroyed immediately thereafter. When the container is created, ffmpeg is executed and listens on a Transmission Control Protocol (TCP) port, through which it streams 30 minutes of video. A native service instance is a single instance of the ffmpeg process; it follows the same lifecycle as the containerized instance.

Management of operations is not trivial, even at the minimum load level, as it involves the following steps:

1. Reboot the video server, to obtain a common and reproducible initial state.
2. Wait until the video server quiesces. This is the time required for server power use to fall to the state where the iLO measurement persistently shows baseline 2 usage. Persistence was empirically found to be ascertained 20 minutes after rebooting.
3. Start the power meters for both total and dynamic power, for both the video server and the virtual switch.
4. Wait for a fifteen-minute interval, to capture behaviour before video streaming.
5. Instantiate and start a container carrying the ffmpeg listener, poised for real-time playback with randomized starting point and 30-minute play time.
6. Start a TSDuck client to connect to the container and measure the bitrate, averaged over 5-second intervals.
7. Once 30 minutes of video have been played, destroy the container.
8. Wait for a fifteen-minute interval, to capture behaviour after video streaming.

For several concurrent streams, steps 5 and 6 must be repeated for each one of the additional streams. For the native service instance, step 5 involves the ffmpeg process only and there is no equivalent to step 7.

It seems evident that manual management is highly prone to error and is therefore unsuitable. Automated management using Python scripts and Ansible [29] is employed to handle the **orchestration** of the various roles: power meters, container runtime managers and video clients. This enables the experiment to be scaled out to levels that are well beyond the physical limitations of a single human operator.

VI. RESULTS

Denote:

- mean dynamic power measured by PowerTOP by $\overline{p_{dyn}^{(ptop)}}$
- mean total power measured by the iLO during a time period T_x by $\overline{p^{(iLO)}}([T_x])$.

A. Video server's baseline 1

Figure 5 shows the power used by the video server over an hour period of measurement, post-onset of quiescence. Since the iLO truncates decimals in $[n, n+1)$ to n, then the computation of the mean will count the incidences of 45 W and 44 W and use them as weights to compute a lower limit to the range of values which the average can take. An upper limit is obtained by adding the maximum possible error (equal to 1W) and the mean of the possible range obtained by adding the mean error (0.5W) to the lower limit of the range. Using this premise, the mean power measured by the iLO, under the condition of a quiescent operating system (see Figure 5) is as follows:

$$P_{b_1}^{(video)} = P_{idle}^{f_1} + P_q^{(os)} = \overline{p^{(iLO)}}([10:31:49, 11:37:01]) = 45.4198\ \text{W} \cong 45.4\ \text{W}$$

B. Mitigation of PowerTOP's limitations

PowerTOP captures neither static nor dynamic power used by Hard Disk Drives (HDDs) and Solid-State Disks (SSDs); this was observed and confirmed through discussion with PowerTOP's developers [30]. Indeed, our experiments under baseline 1 conditions show that if PowerTOP is operated in logging mode with HDD as destination, iLO aggregate power use is more than 0.5 W greater on the SUT than the figure obtained while logging to a RAM (Random-Access-Memory) disk (see Figure 6). Average aggregate power use increases to 46.05W, compared with 45.4W (see Section VI-A, above). On the other hand, while logging to RAM disk (under baseline 1 conditions), average aggregate power use only increases to 45.5W (see Figure 7), compared with 45.4W (see Section VI-A, above) when measurements are taken solely through use of the iLO's instrumentation.

C. Video server's baseline 2

The difference in average dynamic power is added to baseline 1, to obtain baseline 2:

$$\Delta\overline{p_{dyn}^{(ptop)}} = \overline{p_{dyn}^{(ptop)}}(baseline\_2) - \overline{p_{dyn}^{(ptop)}}(baseline\_1) = 0.7727 - 0.1851 = 0.5876\ \text{W}$$

$$\therefore P_{b_2}^{(video)} = P_{idle}^{f_2} + P_q^{(os+dockerd+containerd)} = P_{idle}^{f_1} + P_q^{(os)} + \Delta\overline{p_{dyn}^{(ptop)}} = 45.4 + 0.5876 \cong 45.99\ \text{W}$$

This is consistent with the graphical summarization of iLO measurements shown in Figure 8. This baseline (the graph of power against time) is essential to obtain a reproducible start-

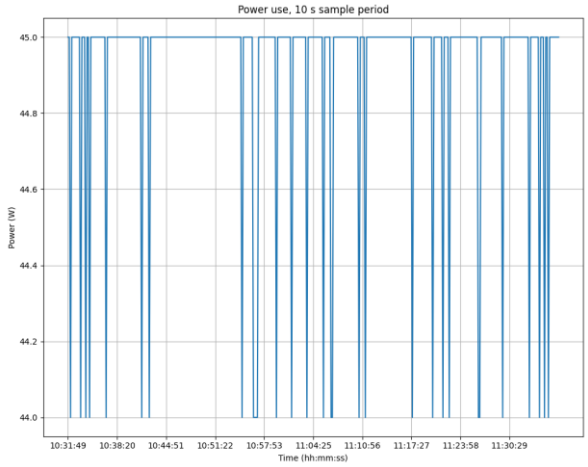


Figure 5. Power used by the video server, with a quiescent OS.

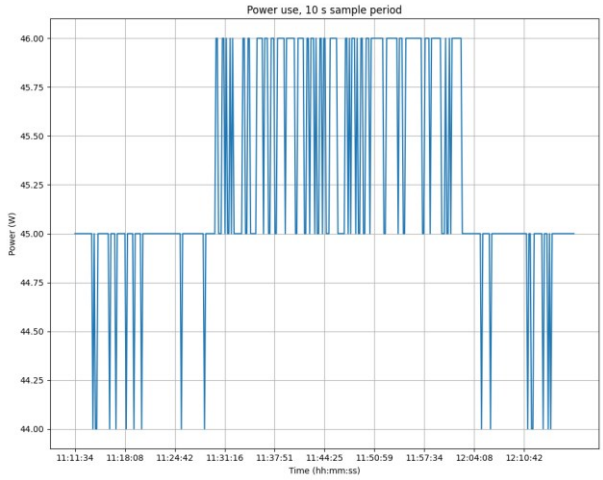


Figure 6. PowerTOP logging [11:30:02,12:02:12] to HDD has a discernable impact on power use

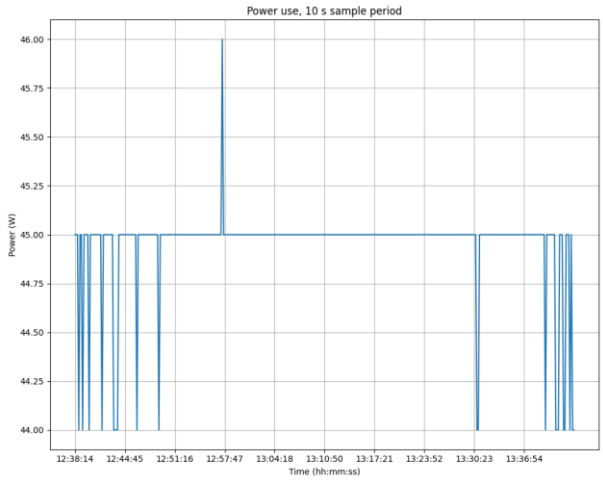


Figure 7. PowerTOP logging [12:57:27,13:29:30] to ramdisk has a lower impact than logging to HDD.

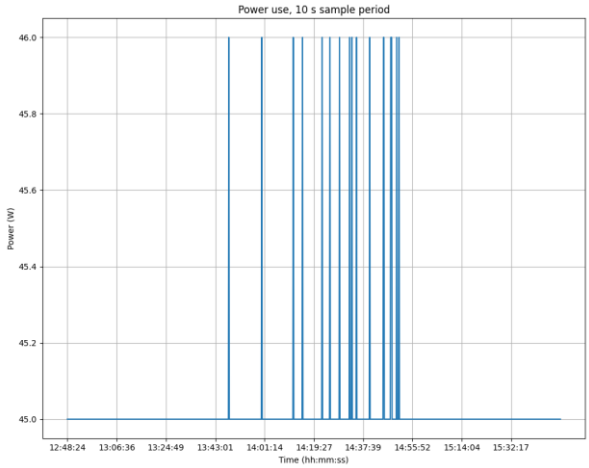


Figure 8. Baseline 2 video server aggregate power

-ing state for all video service operation experiments.

D. *Orchestration of containerized streaming*

Results from running experiments on 1, 2, 5, 10, 20, 40 and 80 instances are presented. The result items consist of:

1. Mean aggregate power use (iLO instrumentation). Due to the integer type of the measurement, actual average iLO power use can lie in the range of ± 0.5 W of the reported result.
2. Mean dynamic power use (PowerTOP instrumentation). Dynamic power data is added to baseline 1 and the sum is plotted on the same Cartesian axes as the total power data.

PowerTOP was used to attribute dynamic power to processes, and these were sorted in descending order. Graphical representations of the power used were produced too. These results are presented in the Github online repository at [31] and in Section VI-F (containerized operation only). Measurements of received stream bitrates are also available in this repository.

1) *Single instance*

Table I shows the mean power use; Figure 9 shows PowerTOP's measurements offset by baseline 1 and laid over the iLO's measurements. Time is shown in the format hh:mm:ss, where hh, mm and ss stand for hour-of-day, minutes in the hour and seconds in the minute, respectively. The larger post-operation (post-op) average power is due to activity undertaken by an instance of containerd (the container runtime) after the container is destroyed (post-ops). However, well after operations end, the iLO's measurements return to the baseline 2 profile. Pre-operations (pre-ops), both meters (iLO and PowerTOP) are in good agreement (PowerTOP's measurements would all be rounded down to 45W). Moreover, the average power used during operations as estimated by PowerTOP is 46.99 W (baseline_1, = 45.4, + 1.5940), whereas the iLO estimates 47.03W. The ten-second

averages' dissimilarity increases during and post-operations but is still good. Notably, the spike in power use at the beginning and end of operations is captured by both meters, albeit not being measurements of the same magnitude.

2) Two instances

Table II and Figure 10 show the results pertinent to two containerized video server instances. As is the case with the single instance, for pre-ops and post-ops, both meters are in good agreement (the spike at about 09:29:00 is probably due to HDD input/output operations while loading PowerTOP). During operations, the average total power estimated by PowerTOP is 48.02 W (baseline_1 + 2.6162), whereas the iLO estimates 47.06W. The discrepancy is an overestimate by about 1W.

An interpretation of the discrepancy between operating period averages is visible in the graph (Figure 8) showing real time measurements. When the iLO measures 46W, the actual value is in the range [46,47), and the rate of change between 46W and 47W is much larger than the single-instance case.

TABLE I. MEAN POWER USE – SINGLE SERVICE INSTANCE

Power type	Description	Avg[a] (W)
$\overline{p^{(iLO)}}[14:47:05,15:03:00]$	Before starting the service instance	45.65
$\overline{p^{(iLO)}}[15:03:00,15:33:05]$	During the service instance's operation	47.03
$\overline{p^{(iLO)}}[15:33:05,15:52:17]$	After the service instance ended	46.17
$\overline{p_{dyn}^{(ptop)}}[14:48:17,15:03:00]$	Mean dynamic power before service instance operation	0.8593
$\overline{p_{dyn}^{(ptop)}}[15:03:00,15:33:05]$	Mean dynamic power during service instance operation	1.5940

a. Average.

TABLE II. MEAN POWER USE – TWO SERVICE INSTANCES

Power type	Description	Avg (W)
$\overline{p^{(iLO)}}[09:24:01,09:44:27]$	Before starting the service instance	45.60
$\overline{p^{(iLO)}}[09:44:27,10:14:37]$	During the service instance's operation	47.06
$\overline{p^{(iLO)}}[10:14:37,10:29:23]$	After the service instance ended	46.12
$\overline{p_{dyn}^{(ptop)}}[09:29:27,09:44:27]$	Mean dynamic power before the service instances' operation	0.9693
$\overline{p_{dyn}^{(ptop)}}[09:44:27,10:14:37]$	Mean dynamic power during the service instances' operation	2.6162

PowerTOP's real time measurements are consistently higher than 47W, revealing that several of the 10-second measurement intervals are in certain disagreement, albeit small (< 2/46, i.e., < 5%).

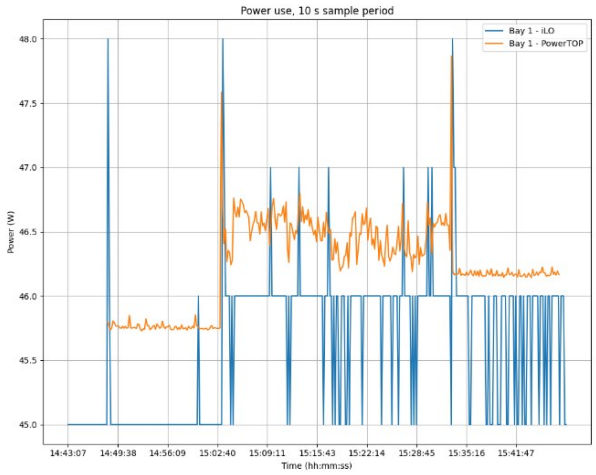


Figure 9. One instance. Video server's power use during containerized service operation. Baseline 1 added to powertop measurements.

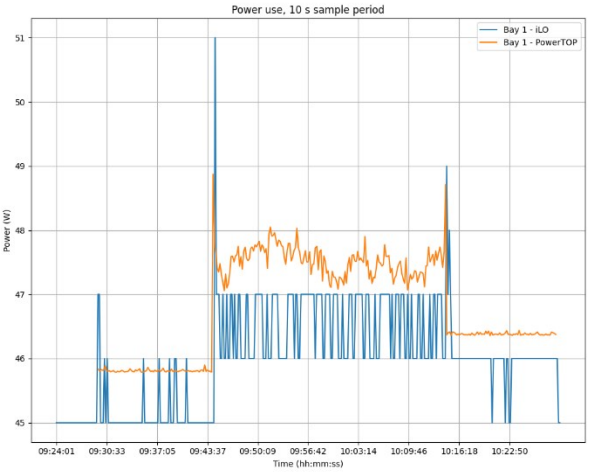


Figure 10. Two instances. Video server's power use during containerized service operation. Baseline 1 added to powertop measurements.

3) Five, ten, twenty, forty and eighty instances

The results for five (Table III, Figure 11), ten (Table IV, Figure 12), twenty (Table V, Figure 13), forty (Table VI, Figure 14) and eighty instances (Table VII, Figure 15) are shown below.

Conditions pre-operations are similar, but PowerTOP's average error estimation increases as power use increases. The numbers shown in the list are PowerTOP's estimate vs iLO's maximum estimate, for N instances (Ni):

- 5i: 50.14 vs 48.66W
- 10i: 54.38 vs 50.10W
- 20i: 60.77 vs 51.74W
- 40i: 64.59 vs 53.90W
- 80i: 60.92 vs 56.80W

Inspection of the online supplementary data on process – level power attribution suggests that PowerTOP overestimates across all processes on our test platform.

TABLE III. MEAN POWER USE – FIVE SERVICE INSTANCES

Power type	Description	Avg, (W)
$\overline{p^{(iLO)}}[11:29:41,11:49:59]$	Before starting the service instance	45.79
$\overline{p^{(iLO)}}[11:49:59,12:20:15]$	During the service instance's operation	48.16
$\overline{p_{dyn}^{(ptop)}}[11:35:00,11:49:59]$	Mean dynamic power before service instances' operation	0.9970
$\overline{p_{dyn}^{(ptop)}}[11:49:59,12:20:15]$	Mean dynamic power during the service instances' operation	4.7421

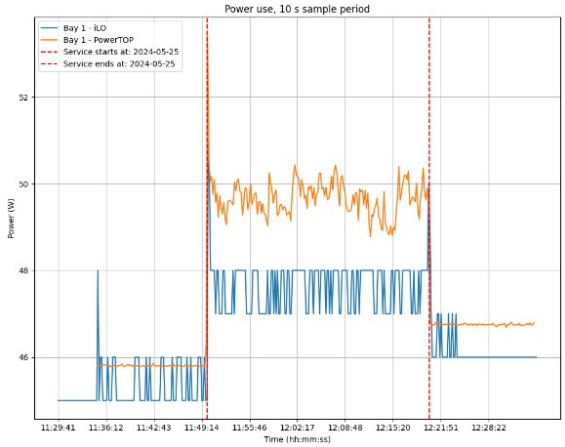


Figure 11. Five instances, containerized operations, baseline 1.

TABLE IV. MEAN POWER USE – TEN SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[15:06:49,15:27:03]$	Before starting the service instance	45.60
$\overline{p^{(iLO)}}[15:27:03,15:57:27]$	During the service instance's operation	49.60
$\overline{p_{dyn}^{(ptop)}}[15:12:03,15:27:03]$	Mean dynamic power before service instances' operation	0.8759
$\overline{p_{dyn}^{(ptop)}}[15:27:03,15:57:27]$	Mean dynamic power during the service instances' operation	8.9781

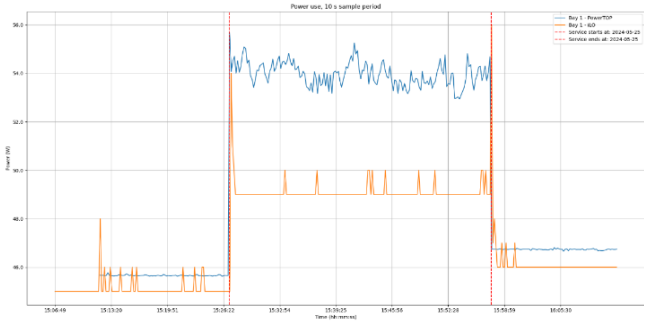


Figure 12. Ten instances, containerized operations, baseline 1.

TABLE V. MEAN POWER USE – TWENTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[17:56:58,18:17:08]$	Before starting the service instance	45.76
$\overline{p^{(iLO)}}[18:17:08,18:48:00]$	During the service instance's operation	51.24
$\overline{p_{dyn}^{(ptop)}}[18:02:08,18:17:08]$	Mean dynamic power before service instances' operation	0.8913
$\overline{p_{dyn}^{(ptop)}}[18:17:08,18:48:00]$	Mean dynamic power during the service instances' operation	15.3720

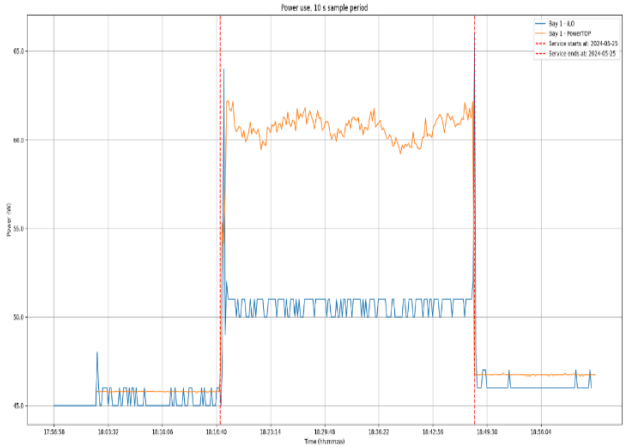


Figure 13. Twenty instances, containerized operations, baseline 1.

TABLE VI. MEAN POWER USE – FORTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[12:57:37,13:17:40]$	Before starting the service instance	45.56
$\overline{p^{(iLO)}}[13:17:40,13:49:15]$	During the service instance's operation	53.40
$\overline{p_{dyn}^{(ptop)}}[13:02:42,13:17:40]$	Mean dynamic power before service instances' operation	0.7206
$\overline{p_{dyn}^{(ptop)}}[13:17:40,13:49:15]$	Mean dynamic power during the service instances' operation	19.1873

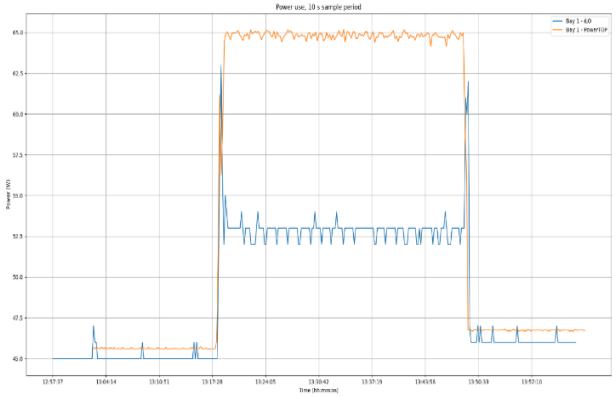


Figure 14. Forty instances, containerized operations, baseline 1

TABLE VII. MEAN POWER USE – EIGHTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[18:19:21,18:39:20]$	Before starting the service instance	45.53
$\overline{p^{(iLO)}}[18:39:30,19:13:53]$	During the service instance's operation	56.30
$\overline{p_{dyn}^{(ptop)}}[18:24:31,18:39:30]$	Mean dynamic power before service instances' operation	0.7435
$\overline{p_{dyn}^{(ptop)}}[18:39:30,19:13:53]$	Mean dynamic power during the service instances' operation	15.5243

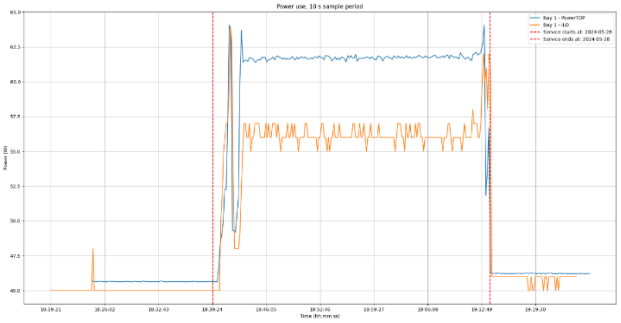


Figure 15. Eighty instances, containerized operations, baseline 1.

E. Orchestation of native streaming

A similar set of experiments was run for native video servers. The results are presented in this section, and are structured in the same manner as that used in Section VI-D.

1) Single instance

TABLE VIII. MEAN POWER USE – ONE SERVICE INSTANCE

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[13:17:42,13:37:55]$	Before starting the service instance	45.54
$\overline{p^{(iLO)}}[13:37:55,14:08:02]$	During the service instance's operation	46.38
$\overline{p_{dyn}^{(ptop)}}[13:22:55,13:37:55]$	Mean dynamic power before service instances' operation	0.2080
$\overline{p_{dyn}^{(ptop)}}[13:37:55,14:08:02]$	Mean dynamic power during the service instances' operation	0.8675

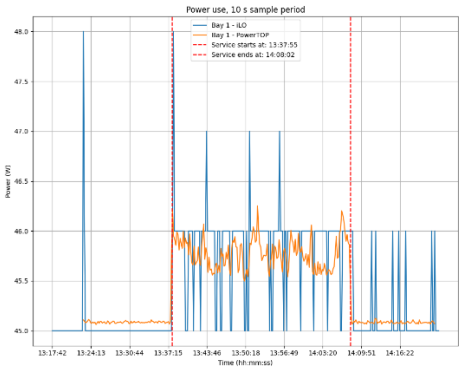


Figure 16. One instance, native operation, baseline 1

2) Two instances

TABLE IX. MEAN POWER USE – TWP SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[15:28:42,15:48:48]$	Before starting the service instance	45.525
$\overline{p^{(iLO)}}[15:48:48,16:18:56]$	During the service instance's operation	46.79
$\overline{p_{dyn}^{(ptop)}}[15:33:47,15:48:48]$	Mean dynamic power before service instances' operation	0.2390
$\overline{p_{dyn}^{(ptop)}}[15:48:48,16:18:56]$	Mean dynamic power during the service instances' operation	1.6653

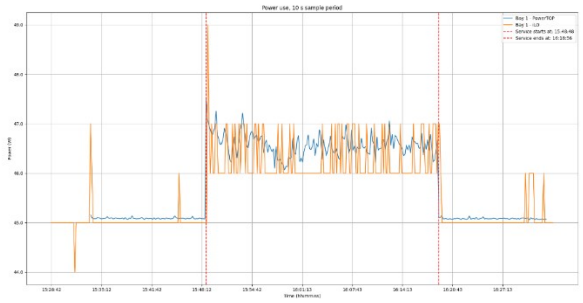


Figure 17. Two instances, native operation, baseline 1

3) Five, ten, twenty, forty and eighty instances

The results for five (Table X, Figure 18), ten (Table XI, Figure 19), twenty (Table XII, Figure 20), forty (Table XIII, Figure 21) and eighty instances (Table XIV, Figure 22) are shown below.

TABLE X. MEAN POWER USE – FIVE SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[17:49:44,18:09:58]$	Before starting the service instance	45.5
$\overline{p^{(iLO)}}[18:09:58,18:40:10]$	During the service instance's operation	47.71
$\overline{p_{dyn}^{(ptop)}}[17:54:57,18:09:58]$	Mean dynamic power before service instances' operation	0.2546
$\overline{p_{dyn}^{(ptop)}}[18:09:58,18:40:10]$	Mean dynamic power during the service instances' operation	3.7906

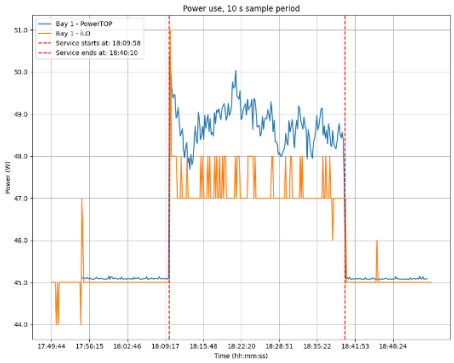


Figure 18. Five instances, native operation, baseline 1

TABLE XI. MEAN POWER USE – TEN SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[19:21:06, 19:41:12]$	Before starting the service instance	45.54
$\overline{p^{(iLO)}}[19:41:12, 20:11:32]$	During the service instance's operation	49.33
$\overline{p_{dyn}^{(ptop)}}[19:26:11, 19:41:12]$	Mean dynamic power before service instances' operation	0.2466
$\overline{p_{dyn}^{(ptop)}}[19:41:12, 20:11:32]$	Mean dynamic power during the service instances' operation	7.4315

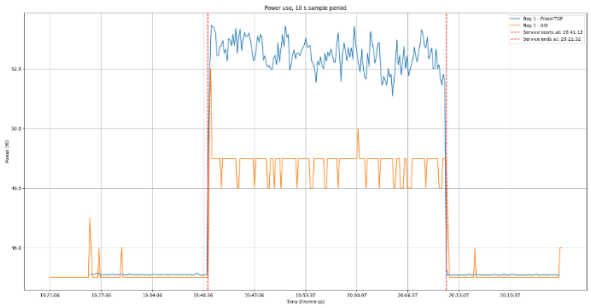


Figure 19. Ten instances, native operation, baseline 1

TABLE XII. MEAN POWER USE – TWENTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[20:52:00, 21:12:09]$	Before starting the service instance	45.52
$\overline{p^{(iLO)}}[21:12:09, 21:42:48]$	During the service instance's operation	51.27
$\overline{p_{dyn}^{(ptop)}}[20:57:09, 21:12:09]$	Mean dynamic power before service instances' operation	0.1819
$\overline{p_{dyn}^{(ptop)}}[21:12:09, 21:42:48]$	Mean dynamic power during the service instances' operation	14.3948

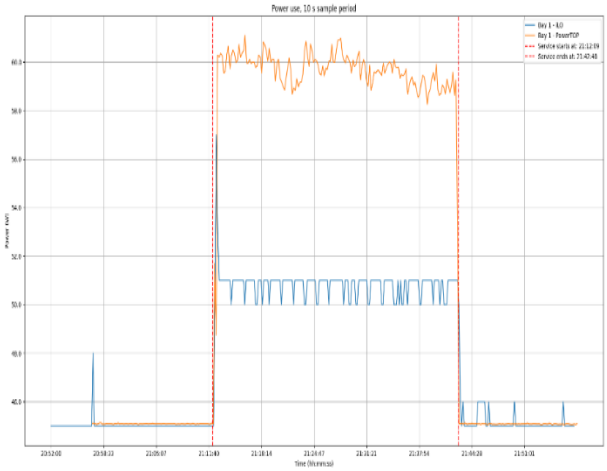


Figure 20. Twenty instances, native operation, baseline 1

TABLE XIII. MEAN POWER USE – FORTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[22:26:05, 22:46:20]$	Before starting the service instance	45.54
$\overline{p^{(iLO)}}[22:46:20, 23:17:43]$	During the service instance's operation	53.27
$\overline{p_{dyn}^{(ptop)}}[22:31:19, 22:46:20]$	Mean dynamic power before service instances' operation	0.1780
$\overline{p_{dyn}^{(ptop)}}[22:46:20, 23:17:43]$	Mean dynamic power during the service instances' operation	19.2853

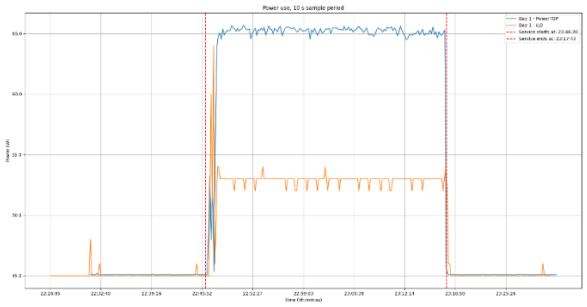


Figure 21. Forty instances, native operation, baseline 1

TABLE XIV. MEAN POWER USE – EIGHTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[00:08:35, 00:28:47]$	Before starting the service instance	45.52
$\overline{p^{(iLO)}}[00:28:47, 01:02:39]$	During the service instance's operation	55.81
$\overline{p_{dyn}^{(ptop)}}[00:13:46, 00:28:47]$	Mean dynamic power before service instances' operation	0.1874
$\overline{p_{dyn}^{(ptop)}}[00:28:47, 01:02:39]$	Mean dynamic power during the service instances' operation	15.1443

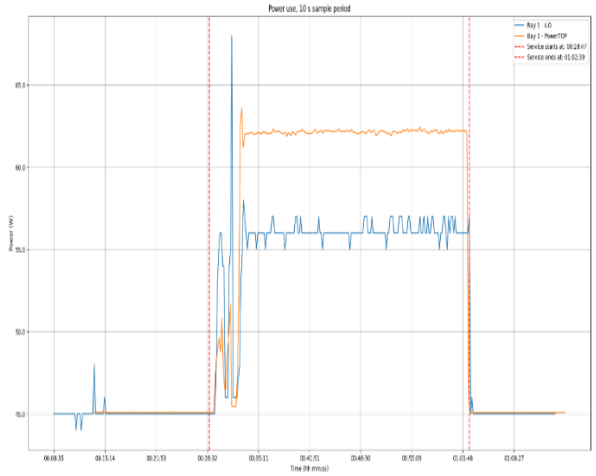


Figure 22. Eighty instances, native operation, baseline 1

F. Breakdown of dynamic power by process, containerized operation

Process power components are sorted in descending order of the mean process power over the period of measurement, until some percentage of the mean total dynamic power over the period of measurement, is obtained. Typical percentages are 50 (the 50th percentile) and 80 (80th percentile). Higher percentiles are avoided, as plots showing the largest power users up to, say, 90% are too dense. Plots and tables are presented to summarize these results.

1) Single instance

Figure 23. 1 instance – process power use, up to 50th percentile

Figure 24. 1 instance – process power use, up to 75th percentile

TABLE XV. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 94TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
[PID 3893] containerd --config /var/run/docker/containerd/containerd.toml	368.73
tick_sched_timer	322.31
[PID 4376] ffmpeg -re -ss 841 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	66.54
[3] net_rx(softirq)	61.61
[PID 3922] containerd --config /var/run/docker/containerd/containerd.toml	59.21
[PID 3920] containerd --config /var/run/docker/containerd/containerd.toml	53.03
[PID 3898] containerd --config /var/run/docker/containerd/containerd.toml	50.45
[PID 3894] containerd --config /var/run/docker/containerd/containerd.toml	40.07
[PID 3907] containerd --config /var/run/docker/containerd/containerd.toml	39.92
toggle_allocation_gate	38.15671

2) Two instances

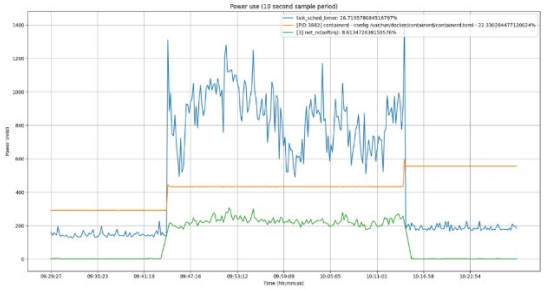


Figure 25. 2 instances – process power use, up to 50th percentile

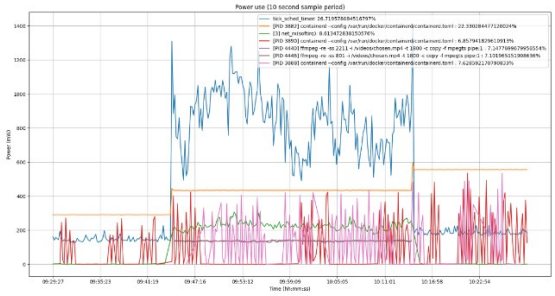


Figure 26. 2 instances – process power use, up to 75th percentile

TABLE XVI. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 87TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	511.72
[PID 3882] containerd --config /var/run/docker/containerd/containerd.toml	427.66
[3] net_rx(softirq)	114.67
[PID 3890] containerd --config /var/run/docker/containerd/containerd.toml	69.27
[PID 4440] ffmpeg -re -ss 2211 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	68.03
[PID 4446] ffmpeg -re -ss 801 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	67.18
[PID 3888] containerd --config /var/run/docker/containerd/containerd.toml	66.81
[PID 3884] containerd --config /var/run/docker/containerd/containerd.toml	64.70
[PID 18] [rcu_preempt]	58.79
[PID 3887] containerd --config /var/run/docker/containerd/containerd.toml	54.53
[PID 3895] containerd --config /var/run/docker/containerd/containerd.toml	42.70
[PID 3897] containerd --config /var/run/docker/containerd/containerd.toml	38.57
toggle_allocation_gate	38.15
[PID 3898] containerd --config /var/run/docker/containerd/containerd.toml	32.94
[PID 3900] containerd --config /var/run/docker/containerd/containerd.toml	22.35

3) *Five instances*

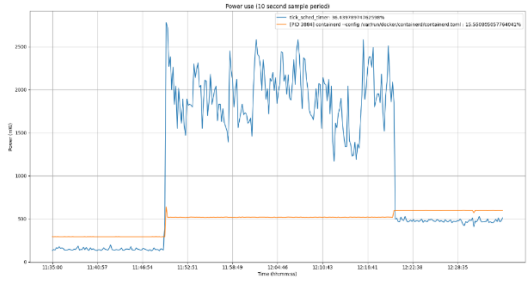


Figure 27. 5 instances – process power use, up to 50th percentile

Figure 28. 5 instances – process power use, up to 80th percentile

TABLE XVII. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 85TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	1123.59
[PID 3884] containerd --config /var/run/docker/containerd/containerd.toml	479.49
[3] net_rx(softirq)	249.06
[PID 17] [rcu_preempt]	99.97
[PID 3886] containerd --config /var/run/docker/containerd/containerd.toml	73.31
[PID 4810] ffmpeg -re -ss 1950 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	67.32
[PID 4808] ffmpeg -re -ss 1629 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	66.53
[PID 4806] ffmpeg -re -ss 2297 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	66.24
[PID 4804] ffmpeg -re -ss 1746 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	66.13
[PID 4812] ffmpeg -re -ss 2792 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	65.24
[PID 4098] containerd --config /var/run/docker/containerd/containerd.toml	60.39
[PID 3895] containerd --config /var/run/docker/containerd/containerd.toml	59.62
[PID 3889] containerd --config /var/run/docker/containerd/containerd.toml	49.85
[PID 3894] containerd --config /var/run/docker/containerd/containerd.toml	47.33
[PID 3893] containerd --config /var/run/docker/containerd/containerd.toml	45.44
[PID 3887] containerd --config /var/run/docker/containerd/containerd.toml	44.97

4) *Ten instances*

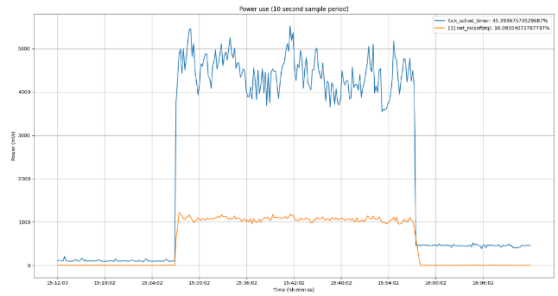


Figure 29. 10 instances – process power use, up to 50th percentile

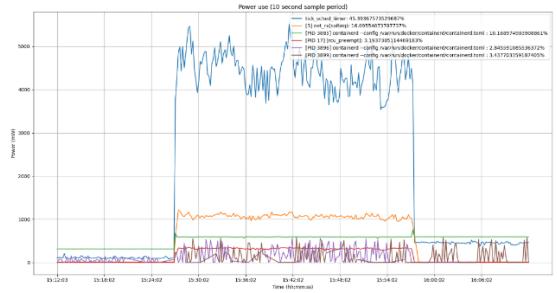


Figure 30. 10 instances – process power use, up to 80th percentile

TABLE XVIII. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 83RD PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	2340.38
[3] net_rx(softirq)	528.77
[PID 3885] containerd --config /var/run/docker/containerd/containerd.toml	524.29
[PID 17] [rcu_preempt]	164.66
[PID 3896] containerd --config /var/run/docker/containerd/containerd.toml	83.20
[PID 3899] containerd --config /var/run/docker/containerd/containerd.toml	76.19
[PID 4102] containerd --config /var/run/docker/containerd/containerd.toml	72.04
[PID 5403] ffmpeg -re -ss 589 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	65.28
[PID 5400] ffmpeg -re -ss 202 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	65.14
[PID 5407] ffmpeg -re -ss 2453 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.91
[PID 5411] ffmpeg -re -ss 121 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.74
[PID 5395] ffmpeg -re -ss 1012 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.58
[PID 5401] ffmpeg -re -ss 235 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.48
[PID 5397] ffmpeg -re -ss 2679 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.42
[PID 5405] ffmpeg -re -ss 1856 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.37

5) Twenty instances

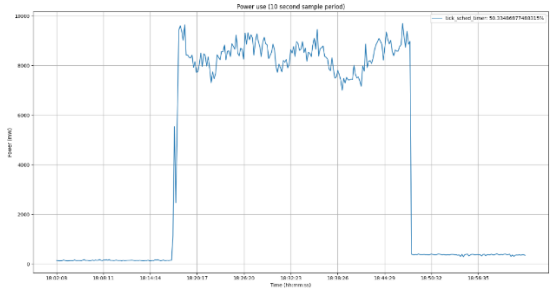


Figure 31. 20 instances – process power use, up to 50th percentile

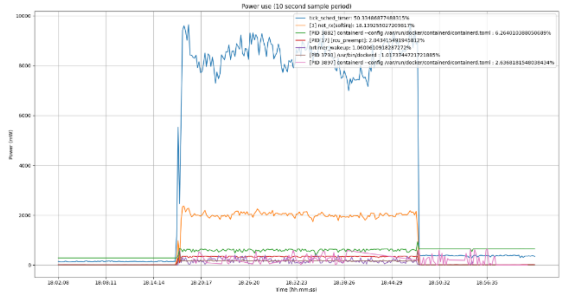


Figure 32. 20 instances – process power use, up to 80th percentile

TABLE XIX. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 78TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	4230.06
[3] net_rx(softirq)	980.63
[PID 3882] containerd --config /var/run/docker/containerd/containerd.toml	526.42
[PID 17] [rcu_preempt]	171.73
hrtimer_wakeup	89.09
[PID 3793] /usr/bin/dockerd	85.50
[PID 3897] containerd --config /var/run/docker/containerd/containerd.toml	72.29
[PID 6622] ffmpeg -re -ss 2972 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	61.28
[PID 6542] ffmpeg -re -ss 2862 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.94
[PID 6592] ffmpeg -re -ss 893 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.90
[PID 6586] ffmpeg -re -ss 990 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.68
[PID 6550] ffmpeg -re -ss 1550 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.62
[PID 6562] ffmpeg -re -ss 99 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.57
[PID 6616] ffmpeg -re -ss 924 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.53
[PID 6610] ffmpeg -re -ss 2989 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.52

6) Forty instances

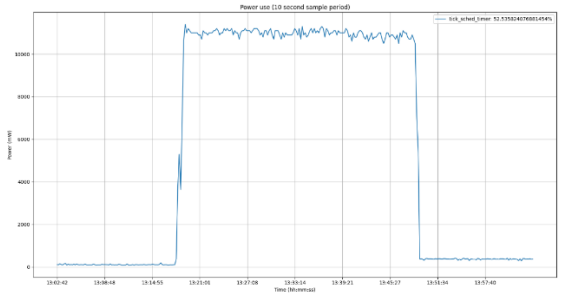


Figure 33. 40 instances – process power use, up to 50th percentile

Figure 34. 40 instances – process power use, up to 80th percentile

TABLE XX. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 80TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	5402.63
[3] net_rx(softirq)	1265.36
hrtimer_wakeup	663.73
[PID 3882] containerd --config /var/run/docker/containerd/containerd.toml	405.74
[PID 18] [rcu_preempt]	104.85
[PID 3795] /usr/bin/dockerd	100.95
[PID 3899] containerd --config /var/run/docker/containerd/containerd.toml	44.12
[PID 3884] containerd --config /var/run/docker/containerd/containerd.toml	42.31
[PID 3887] containerd --config /var/run/docker/containerd/containerd.toml	40.81
[PID 7927] ffmpeg -re -ss 1413 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.60
[PID 7969] ffmpeg -re -ss 237 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.52
[PID 3898] containerd --config /var/run/docker/containerd/containerd.toml	36.22
[PID 7933] ffmpeg -re -ss 60 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.21
[PID 7993] ffmpeg -re -ss 2988 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.15
[PID 7913] ffmpeg -re -ss 1714 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.07

7) Eighty instances

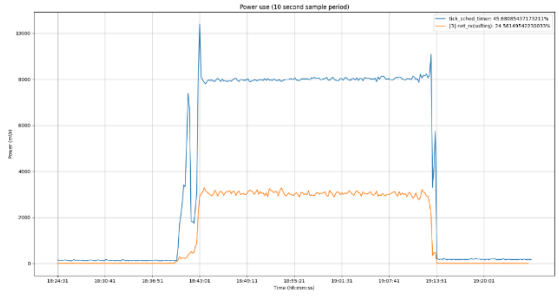


Figure 35. 80 instances – process power use, up to 50th percentile

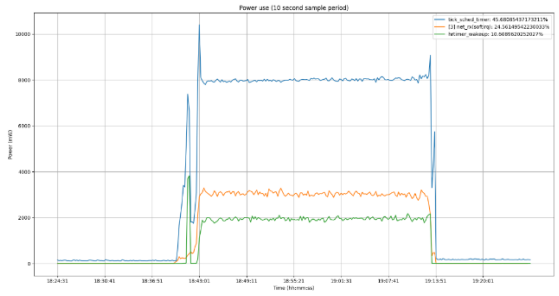


Figure 36. 80 instances – process power use, up to 80th percentile

TABLE XXI. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 79TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	4038.35
[3] net_rx(softirq)	1449.76
hrtimer_wakeup	935.01
[PID 3882] containerd --config /var/run/docker/containerd/containerd.toml	251.19
[PID 3794] /usr/bin/dockerd	103.59
[PID 18] [rcu_preempt]	68.98
[PID 3881] containerd --config /var/run/docker/containerd/containerd.toml	37.20
[PID 3892] containerd --config /var/run/docker/containerd/containerd.toml	29.13
toggle_allocation_gate	28.56
[PID 3897] containerd --config /var/run/docker/containerd/containerd.toml	24.14
[7] sched(softirq)	23.67
[PID 8262] ffmpeg -re -ss 1711 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	20.19
[PID 3893] containerd --config /var/run/docker/containerd/containerd.toml	19.72
[PID 9089] ffmpeg -re -ss 247 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	19.58
[PID 11433] ffmpeg -re -ss 415 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	19.02

VII. ANALYSIS

Various characterizations of power use are considered and plotted in Figure 37. In the notation shown below, the (n) symbol indicates dependence of power used on number of streaming containers.

1. total power during operations, $P_{ops}^{iLO}(n)$, and
2. differential total power, where the difference is between operations and quiescence, $P_{ops}^{iLO}(n) - P_{q}^{iLO}$.

Figure 37 illustrates the results in graphical form. The top row of graphs compares total power and differential total power, respectively, for containerized and native operations. The bottom row shows the difference between total power and differential total power. The non-monotonic behaviour seen in the bottom row is due to the error introduced by the rounding of iLO instrumentation.

$P_{b_1}^{(video)}$ (45.4W) was used as the offset for dynamic power obtained using PowerTOP, at every instance count. The value of $P_{b_2}^{(video)}$ (45.99W), obtained by adding the increment in dynamic power inferred by PowerTOP (see Section VI-C), was larger than that measured as $\overline{p^{(iLO)}}[service\_operation\_time]$ for any of the instance counts. If, however, $P_{b_2}^{(video)}$ is obtained in the same way as $P_{b_1}^{(video)}$, by averaging $p^{(iLO)}$ over the relevant period of time, attainment of $P_{b_2}^{(video)}$ supports well the purpose of good known starting and ending events for a run of experimentation.

Dynamic power measurements as a function of streaming videos are not shown in Figure 37, as PowerTOP's measurements do not produce consistent, intelligible results on our platform. Estimates are insufficiently accurate. PowerTOP is capable of capturing power change behaviour (see, notably, Figure 36), but it requires further development before its estimates can be used for quantitative analysis.

On the other hand, notwithstanding PowerTOP's problematic scaling, its capability to capture power change suggests that its relative attribution of power consumption to processes is sound. Basing upon this understanding, it is possible to detect that, notwithstanding the potential to save power consumption through use of tickless kernels, the tick_sched_timer function is easily the largest power consumer (see Figures 27, 29, 31, 33 and 35).

VIII. CONCLUSION AND FUTURE WORK

The objective set out in Section III was to quantify the overhead incurred by operating the video service containerized, instead of as an application running directly on the host operating system (native operation). An access network of the Active Ethernet type was constructed and a video cache deployed in an access node to stream videos to the access node's service area. An implementation model describing the access network was included.

The results obtained have shown that the overhead is negligible and that the benefit of running the video source in a container comes at little cost. The possibility of

consolidating video streaming containers can be pursued with confidence.

No discernable cause for concern was found in the power measurement instrumentation embedded in the HPE Gen9 platform. Documentation on interfacing with the Integrated Lights-Out (iLO) server management was readily available. For detail beyond typical interest, HPE readily divulged information on this tool when contacted for help, including, for example, the method used to round the power measurement into an integer [32].

On the other hand, PowerTOP's accuracy poses a problem. The various graphs of power against time have shown that it captures changes well but significantly overestimates them. In the light of these errors, works that have investigated containerization's overhead with the use of this tool (e.g., [15]) may need to be reviewed for the implications of inaccuracies introduced by the tool, perhaps by using external, physical power meters to calibrate PowerTOP's measurement.

Baselines have been obtained for both the video server and the virtual switch. In particular, P_{b2}^{video} has been found useful in obtaining a reproducible starting point for experiments; to a lesser extent, P_{b1}^{video} has been found useful in providing an offset for power obtained through tools that measure dynamic power. This segues well into an observation that merits particular attention. Even with 80 concurrent streams, the static power has dwarfed the dynamic power. The importance of this observation pertains to the importance of the benefit of containerization as an enabler of consolidation of physical hosts. It can readily be stated that the overhead incurred in providing the **service framework** of containerization poses no obstacle to exploration of exploitation of this benefit.

PowerTOP's limitations invite researchers to explore its causes, as the demand for software power meters is pressing in multi-tenant hosting. Future work would do well to assess the relative accuracy of PowerTOP with Scaphandre [33] prior to embarking on the use of either within power instrumentation.

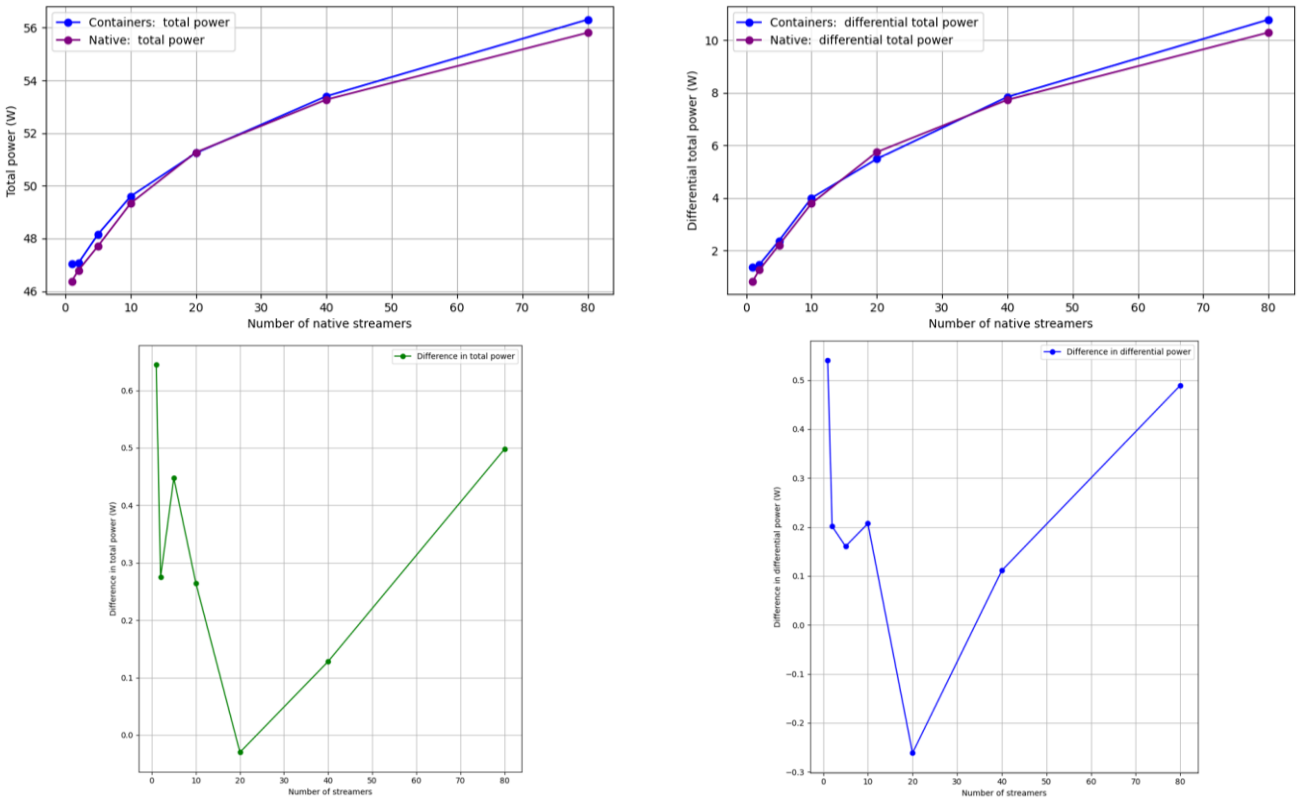


Figure 37. Comparison: native vs containerized streaming. Clockwise from top left: $P_{ops}^{iLO}(n)$, $P_{ops}^{iLO}(n) - P_{q}^{iLO}$, $P_{ops}^{iLO}(n_{cont}) - P_{ops}^{iLO}(n_{native})$ and $\left(P_{ops}^{iLO}(n_{cont}) - P_{q}^{iLO}{}_{cont}\right) - \left(P_{ops}^{iLO}(n_{native}) - P_{q}^{iLO}{}_{native}\right)$.

REFERENCES

[1] E.-V. Depasquale and S. Zammit, 'Containerization's Power Use Overhead in Video Streaming', in Proceedings of the Ninth International Conference on Green Communications, Computing and Technologies (GREEN 2024), Nice, France: IARIA, 2024, pp. 1–9. [Online]. Available: https://www.thinkmind.org/articles/green_2024_1_10_80008.pdf

[2] A. Teker, A. H. Örnek, and B. Canberk, 'Network Bandwidth Usage Forecast in Content Delivery Networks', in 2020 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom), Jul. 2020, pp. 1–6. doi: 10.1109/CoBCom49975.2020.9174180.

[3] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, 'FemtoCaching: Wireless Content Delivery Through Distributed Caching Helpers', IEEE Trans. Inf. Theory, vol. 59, no. 12, pp. 8402–8413, Dec. 2013, doi: 10.1109/TIT.2013.2281606.

[4] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, 'Cache in the air: exploiting content caching and delivery techniques for 5G systems', IEEE Commun. Mag., vol. 52, no. 2, pp. 131–139, Feb. 2014, doi: 10.1109/MCOM.2014.6736753.

[5] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, 'A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications', IEEE Access, vol. 5, pp. 6757–6779, 2017, doi: 10.1109/ACCESS.2017.2685434.

[6] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, 'A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges', IEEE Commun. Surv. Tutor., vol. 20, no. 1, pp. 416–464, 2018, doi: 10.1109/COMST.2017.2771153.

[7] J. Zhao, P. Liang, W. Liufu, and Z. Fan, 'Recent Developments in Content Delivery Network: A Survey', in Parallel Architectures, Algorithms and Programming, H. Shen and Y. Sang, Eds., Singapore: Springer, 2020, pp. 98–106. doi: 10.1007/978-981-15-2767-8_9.

[8] B. Zolfaghari et al., 'Content Delivery Networks: State of the Art, Trends, and Future Roadmap', ACM Comput. Surv., vol. 53, no. 2, p. 34:1-34:34, Apr. 2020, doi: 10.1145/3380613.

[9] S. Rivoire, P. Ranganathan, and C. Kozyrakis, 'A comparison of high-level full-system power models', in Proceedings of the 2008 conference on Power aware computing and systems, in HotPower'08. USA: USENIX Association, Dec. 2008, p. 3.

[10] E.-V. Depasquale, F. Davoli, and H. Rajput, 'Dynamics of Research into Modeling the Power Consumption of Virtual Entities Used in the Telco Cloud', Sensors, vol. 23, no. 1, Art. no. 1, Jan. 2023, doi: 10.3390/s23010255.

[11] K. Hinton, F. Jalali, and A. Matin, 'Energy consumption modelling of optical networks', Photonic Netw. Commun., vol. 30, no. 1, pp. 4–16, Aug. 2015, doi: 10.1007/s11107-015-0491-5.

[12] W. Lin, H. Wang, and W. Wu, 'A power monitoring system based on a multi-component power model', Int. J. Grid High Perform. Comput., vol. 10, no. 1, pp. 16–30, Jan. 2018.

[13] B. Rountree, D. K. Lowenthal, M. Schulz, and B. R. de Supinski, 'Practical performance prediction under Dynamic Voltage Frequency Scaling', in 2011 International Green Computing Conference and Workshops, Jul. 2011, pp. 1–8. doi: 10.1109/IGCC.2011.6008553.

[14] T. Guérout, Y. Gaoua, C. Artigues, G. Da Costa, P. Lopez, and T. Monteil, 'Mixed integer linear programming for quality of service optimization in Clouds', Future Gener. Comput. Syst., vol. 71, pp. 1–17, Jun. 2017, doi: 10.1016/j.future.2016.12.034.

[15] R. Bolla, R. Bruschi, F. Davoli, C. Lombardo, and N. S. Martinelli, 'Analyzing the Power Consumption in Cloud-Native 5/6G Ecosystems', in 2023 IEEE International Conference on Communications Workshops (ICC Workshops), May 2023, pp. 611–617. doi: 10.1109/ICCWorkshops57953.2023.10283755.

[16] Environmental Engineering (EE); Green Abstraction Layer (GAL); Power management capabilities of the future energy telecommunication fixed network nodes; Enhanced Interface for power management in Network Functions Virtualisation (NFV) environments, Sophia Antipolis, France., Jan. 2024.

[17] Multi-service Broadband Network Architecture and Nodal Requirements, TR-178 Issue 2, Broadband Forum., Sep. 2017. [Online]. Available: https://www.broadband-forum.org/technical/download/TR-178_Issue-2.pdf

[18] Hewlett Packard Enterprise, 'HPE ProLiant BL460c Gen9 Server Blade', PSNow. Accessed: Jun. 17, 2024. [Online]. Available: https://www.hpe.com/psnow/doc/c04347343

[19] 'FFmpeg'. Accessed: Jun. 17, 2024. [Online]. Available: https://ffmpeg.org/

[20] 'TSDuck'. Accessed: Jun. 17, 2024. [Online]. Available: https://tsduck.io/

[21] 'Open vSwitch'. Accessed: Jun. 17, 2024. [Online]. Available: https://www.openvswitch.org/

[22] 'index | Alpine Linux'. Accessed: Jun. 17, 2024. [Online]. Available: https://alpinelinux.org/

[23] 'Docker: Accelerated Container Application Development'. Accessed: Jun. 17, 2024. [Online]. Available: https://www.docker.com/

[24] 'Kubernetes Documentation', Accessed: Sep. 03, 2019. [Online]. Available: https://kubernetes.io/docs/home/

[25] DMTF Redfish Forum, Redfish Specification, DSP0266, Apr. 03, 2024. Accessed: Jun. 17, 2024. [Online]. Available: https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.20.1.pdf

[26] 'Powertop - ArchWiki'. Accessed: Feb. 20, 2024. [Online]. Available: https://wiki.archlinux.org/title/powertop

[27] A. van de Ven, 'PowerTOP running average interval and display update interval', Apr. 08, 2024.

[28] A. van de Ven, 'PowerTOP running average interval and display update interval (update)', Apr. 08, 2024.

[29] Ansible Community Documentation, 'User Guide'. Accessed: Jun. 17, 2024. [Online]. Available: https://docs.ansible.com/ansible/latest/user_guide/index.html

[30] A. van de Ven, 'PowerTOP running average interval and display update interval (2nd update)', Apr. 30, 2024.

[31] edepa, edepa/video_streaming_power_use. (Jun. 18, 2024). Accessed: Mar. 15, 2025. [Online]. Available: https://github.com/edepa/video_streaming_power_use

[32] J. Sultana, 'Power measurement - Gen9', May 24, 2024.

[33] 'Introduction - Scaphandre documentation'. Accessed: Mar. 15, 2025. [Online]. Available: https://hubblo-org.github.io/scaphandre/book/

Structural Health Monitoring of Steel Tower Structure Using Long-term Vibration Sensing with High-precision Accelerometers and MEMS Accelerometers

Narito Kurata
Faculty of Industrial Technology
Tsukuba University of Technology
Tsukuba City, Ibaraki, Japan
e-mail: kurata@home.email.ne.jp

Abstract - **Many steel tower structures built in Japan are aging and are increasingly at risk of collapse in the event of typhoons or major earthquakes. At the same time, it is difficult to proceed with rebuilding such aging structures with limited budgets and manpower, thus there is a need to monitor the health of the structures, and to equalize and facilitate the rebuilding work. For this purpose, it is necessary to make full use of sensing technology to diagnose deterioration in order to accurately assess the condition of steel tower structures. In this study, sensors were installed at multiple locations on an actual steel tower structure, and measurements were taken over a period of nine months. Data was measured and saved for five minutes, centered on the time when the maximum Root Mean Square (RMS) value of acceleration occurred each day. There have been no other studies in which measurements were taken and analyzed at the time when the maximum vibration occurred each day, with almost no missing data, over several months of constant measurement. This study aims to obtain knowledge that will be useful for monitoring the health of structures and for leveling and smoothing out reconstruction work in order to prevent loss of life due to the collapse of steel tower structures that have not yet been made apparent. In addition, although the use of high-precision accelerometers is preferable for data analysis, it is confirmed that the use of inexpensive MEMS accelerometers is sufficient in actual practice.**

Keywords-Vibration Sensing; Steel Tower Structure; Structural Health Monitoring; Micro Electro Mechanical Systems; Frequency Analysis

I. INTRODUCTION

In Japan, the civil infrastructure is aging, with approximately 60% of road bridges, river management facilities, such as sluice gates, and port quays, among them, being more than 50 years old after construction by 2033 [1]. This situation has been pointed out since the early 2000s, but it came to be highlighted as an important social issue after the 2013 Sasago Tunnel ceiling plate fall accident, in which nine people lost their lives when the vehicles they were traveling in became trapped under the fallen ceiling [2]. In a similar context, the maintenance and reconstruction of steel tower structures, which are ubiquitous throughout Japan for communication and power transmission, is one of the most pressing social issues that has yet to be acknowledged. Steel tower structures consist of a steel framework, and are used for power transmission towers, communication towers for mobile phone base stations, and transmission towers for broadcasting stations, as well as watchtowers for weather observations, lighthouses and firefighting. For power transmission alone, there are currently 240,000 towers supporting the electricity supply [3], and along with communication towers, they form the very infrastructure that underpins our social life. The rush to build transmission towers began in the 1970s, and although the average service life of these towers is estimated to be around 40 years, 120,000, or about half of the total, are overdue for renewal. For example, Typhoon No. 15, which hit the Japanese archipelago in September 2019, caused material damage, including the collapse of a transmission tower in Kimitsu, Chiba Prefecture, which was built in the 1970s, but fortunately no human casualties were reported [4]. For this reason, the importance of this social issue, as regards the maintenance and management of infrastructure as tunnels and bridges, has not yet been recognized. Clearly, many steel towers have reached the point where they need renewal, and the risk of collapse in the event of a typhoon or major earthquake is increasing. It is difficult, however, to renew of all of them at once with limited budgets and manpower, there is an urgent need to monitor their health and extend their service life, as well as to equalize and facilitate reconstruction work. It is therefore essential to accurately assess the condition of steel towers, and to diagnose deterioration using sensing and digital technology.

We have been researching and developing sensing systems that achieve highly accurate autonomous time synchronization for structural health monitoring and earthquake observation. In the light of the current situation, this study collected and analyzed measurement data over several months by installing sensors on a real steel tower in service to obtain knowledge useful for monitoring the health of towers and extending their service life, as well as for equalizing and facilitating reconstruction work, thereby to prevent loss of human life due to unforeseen tower collapse [1].

Section II describes the state of the art of the research for structural health monitoring and sensor technologies. Section III describes the targeted steel tower structure and the installed sensing system, while Section IV presents the results of an analysis of the measured data. Section V gives a comparison of the measurement results obtained by high-precision accelerometers and MEMS accelerometers, and a discussion of their usefulness. From these results, Section VI presents the conclusions of the study and future tasks.

II. STATE OF THE ART

Many studies on the health monitoring of structures have been conducted since the early 2000s, and papers summarizing research trends at that time have been published [5]. Techniques and methodologies for monitoring of structures have been summarized, focusing on the modeling of structural behavior, data analysis methods and sensor technologies [6][7]. Many of the studies focus on social infrastructure structures, such as bridges and highways, where inspections are mandatory, and technologies, such as fiber-optic sensors, wireless sensor networks and image processing have been applied [8][9]. In recent years, machine learning has been utilized in this field, and efforts have been made to apply machine learning algorithms, acquire and process data, and improve prediction capabilities [10]. Moreover, real-time monitoring, data analysis and optimization of maintenance planning have been carried out with regard to structural health monitoring and management systems based on the Internet of Things (IoT) and cloud computing [11]. However, for steel tower structures, inspections are not mandatory and few studies have focused on maintenance management through structural health monitoring. In addition, because transmission towers are owned by electric power companies while communications towers are owned by telecommunications carriers, information on initiatives targeting them is not publicly available. In this study, sensors were installed at multiple locations on an actual steel tower structure, and measurements were taken over a period of nine months. Data was measured and saved for five minutes, centered on the time when the maximum RMS value of acceleration occurred each day. There have been no other studies in which measurements were taken and analyzed at the time when the maximum vibration occurred each day, with almost no missing data, over several months of constant measurement [6][7][8]. This study aims to obtain knowledge that will be useful for monitoring the health of structures and for leveling and smoothing out reconstruction work in order to prevent loss of life due to the collapse of steel tower structures that have not yet been made apparent.

III. STEEL TOWER STRUCTURE AND SENSING SYSTEM

For the many steel towers that require renewal, rather than just focusing on the number of years since construction, it is important to accurately assess their condition in order to monitor the health of individual towers, extend their service life, and equalize and facilitate reconstruction work. To perform such analyses, data must be acquired and collected from actual steel tower structures. Long-term measurements over a period of several months were therefore carried out on a steel tower constructed in Numazu, Shizuoka Prefecture, as shown in Figure 1. Also, it shows the arrangement and types of sensors used. The sensors were placed at the top, center and base of the steel tower in order to obtain data on the overall and local behavior of the tower, which is directly related to any damage it may have sustained.

Table I lists the installation location and the measurement data obtained. Accelerometers measure the acceleration at the top, center and base of the tower. In each part of the tower, both high-precision accelerometers and inexpensive MEMS accelerometers were installed. Table II lists the specifications of the two types of accelerometers. The high-precision accelerometer has a 3-axis crystal acceleration sensor with high accuracy and excellent stability, which is micro-fabricated from a highly accurate and stable crystal material. As shown in Table II, it has low noise and low power consumption and is capable of high-resolution vibration measurement. Compared to high-precision accelerometer, MEMS accelerometer has lower noise density performance, but it can be operated with low power consumption and can be procured at low cost. Although it is desirable to use high-precision accelerometers for any analysis, in view of the widespread use of measurement systems, it is important to determine the extent to which analysis is possible with MEMS accelerometers, which are not highly accurate but are very inexpensive.

The sampling frequency was 100 Hz, and data was measured and saved for 5 minutes around the time when the maximum RMS value of acceleration occurred on each day. Each sensor module had a built-in battery and wireless communication function that used the 2.4 GHz frequency band, and transmitted data wirelessly to a data logger for data recording, which was installed at the base shown in Figure 1. The batteries installed in each sensor module can power the system for approximately one year without needing to be replaced. Each sensor module stores data in its memory and transmits the data wirelessly to the logger. The logger was installed in a small temporary structure adjacent to the base of the tower structure and provided with Internet access, allowing data to be collected remotely.

TABLE I. INSTALLATION LOCATIONS AND MEASUREMENT DATA OF SENSOR

Sensor	Installation Location	Measurement Data
Accelerometer	Tower top, center, base	Acceleration in two horizontal directions and vertical direction

TABLE II. SPECIFICATIONS OF HIGH-PRECISION AND MEMS ACCELEROMETER

	High-precision Accelerometer	MEMS Accelerometer
Model	EPSON M-A351AS	Analog Devices ADXL355
Direction	3 axes (2 horizontal, 1 vertical)	3 axes (2 horizontal, 1 vertical)
Range	±5G	±2G
Noise Density	0.5 μG/√Hz	22.5 μG/√Hz
Operating Temperature	-20 ℃ to +85 ℃	-40 ℃ to +125 ℃
Power Consumption	20mA	200μA
Interface	SPI	SPI/I^2C

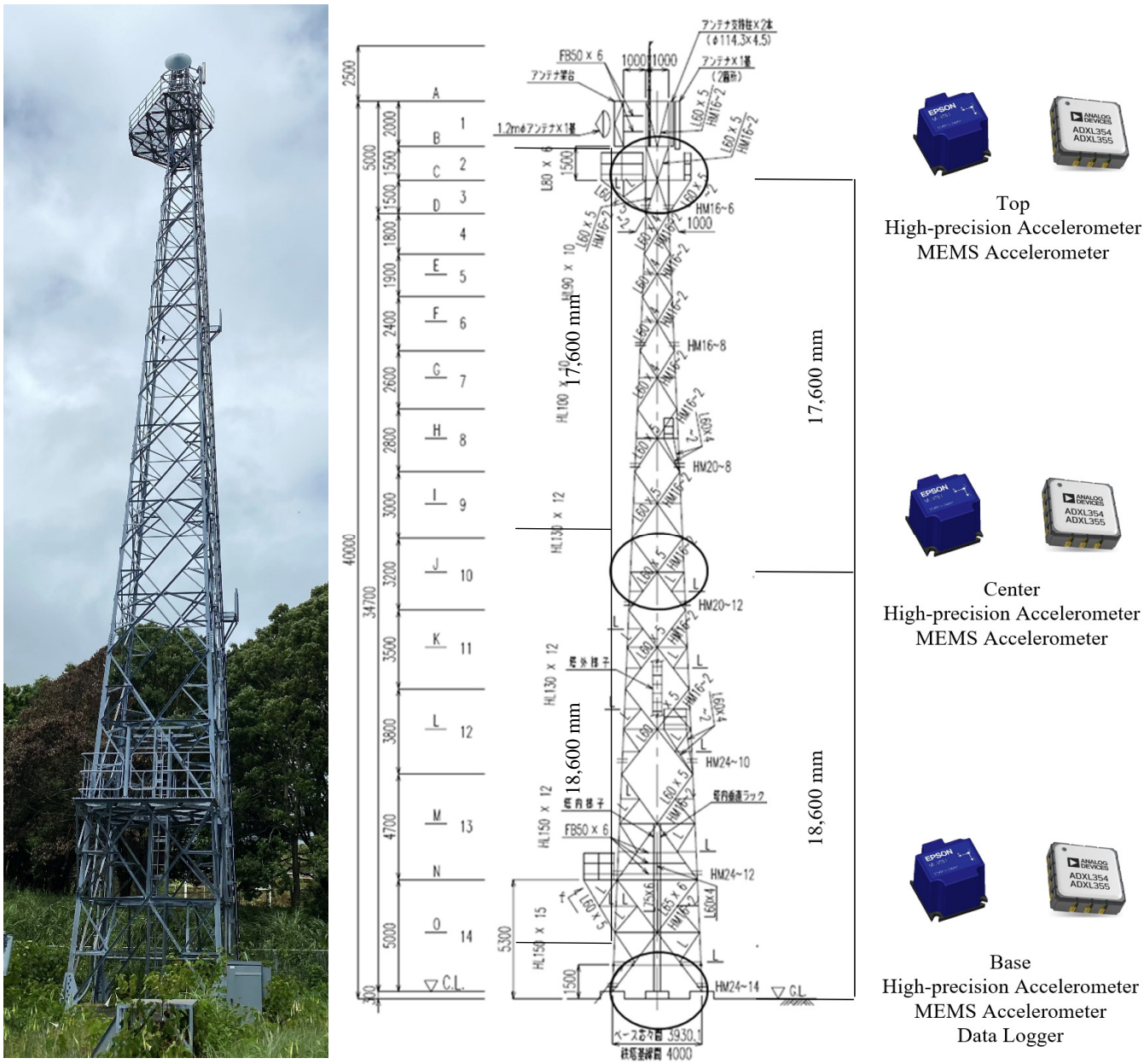


Figure 1. Steel tower structure and sensor location.

IV. Data Analysis

The accelerometers measured the acceleration at the top, center and base to capture the main vibration modes of the steel tower, and to check for excessive vibration and deformation in each part and between parts. In this study, measurements were taken over a nine-month period from February to October 2023. Data was measured and stored for five minutes, centered on the time when the maximum RMS value of daily acceleration occurred. There have been no other research studies in which measurements were taken almost without missing data at the time when the maximum vibration occurred each day on an actual tower structure, and the results of frequency analysis over several months were presented [4][5][6]. The results below show the long-term changes in the dynamic characteristics of the tower structure.

Figure 2 shows the maximum values measured by high-precision accelerometers during the measurement period.

Figure 3 shows the Fourier spectra of the measured data at the top of the tower obtained by a high-precision accelerometer during the measurement period. In Figures 3(a) to (c), the diagrams on the left show the 3D Fourier spectrum with the date and frequency during the nine-month

measurement period on the horizontal axis and amplitude plotted on the vertical axis; the diagrams on the right show this spectrum viewed from directly above, with the date and frequency on the axis and the amplitude in color. The white areas in the right diagrams are areas of missing data. The diagrams on the left give an overview of which frequencies were predominant with respect to acceleration at the top of the tower during the nine-month period, and whether these frequencies changed or not. From the diagrams on the right, the predominant frequencies and their changes can be clearly observed. Since this is the acceleration at the top of the tower, the first-order natural frequencies in the horizontal directions of the targeted steel tower can be obtained. From Figure 3(a), in the x-direction, the horizontal natural frequency can be observed at 1.66 Hz, and from (b), in the y-direction, at 1.59 Hz. Also, no change was observed during the nine months when the measurements were conducted. From (c), the natural frequency in the vertical direction (z-direction) can be observed at 42.5 Hz. Natural frequency is the inherent resonant frequency of a structure. It varies depending on the shape, restraint position, and Young's modulus and density of the material. At this frequency, once an external force is applied, the structure resonates and continues to vibrate on its own without the application of an external force. Natural frequency is determined by the mass and stiffness of the structure, so if the structure is damaged, its stiffness decreases and the natural frequency decreases.

Figure 4 shows the Fourier spectra of the measured data obtained at the center of the tower by the high-precision accelerometer during the measurement period. Since it is the acceleration at the center, the second-order natural frequencies in the horizontal direction of the targeted steel tower can be obtained. From Figures 4(a) and (b), the dominant frequency is around 6.5 Hz in both the x- and y-directions, but there is no clear peak. It can also be seen that there was no change during the nine months in which the measurements were conducted.

The day when the maximum acceleration occurred is indicated by ▼ in Figure 2. Figure 5 shows the time history waveforms of the data measured at the top of the tower by the high-precision accelerometer at this time. From Figure 5, it can be seen that at the top of the tower, accelerations of similar magnitude occur in the x and y-directions, which are horizontal directions, while almost no acceleration occurs in the z-direction, which is the vertical direction. As indicated in (2) above, the natural frequency in the x-direction can be observed at 1.66 Hz, and from (b), the horizontal natural frequency in the y-direction, at 1.59 Hz. In general, there is little structural difference between steel towers in the x- and y-directions, thus the natural frequencies are also close.

Figures 6 and 7 show the time history waveforms and Fourier spectra of the measured data in each direction by the high-precision accelerometers on the day when the maximum acceleration of the measurement period occurred. In both the x-direction shown in Figure 6 and the y-direction shown in Figure 7, the base of tower does not vibrate, and the acceleration increases towards the center and the top of the tower, indicating the first-order mode vibrations are dominant, while the higher-order mode vibrations are small. Therefore, the first-order natural frequencies in two horizontal directions are appropriate as a risk indicator for detecting deterioration and damage to steel towers. If there is a change in the risk index, specifically if a trend toward a decrease in the natural frequency is observed, it can be assumed that the stiffness of the structure has decreased, which will lead to an increased priority for detailed inspections and planned reconstruction work.

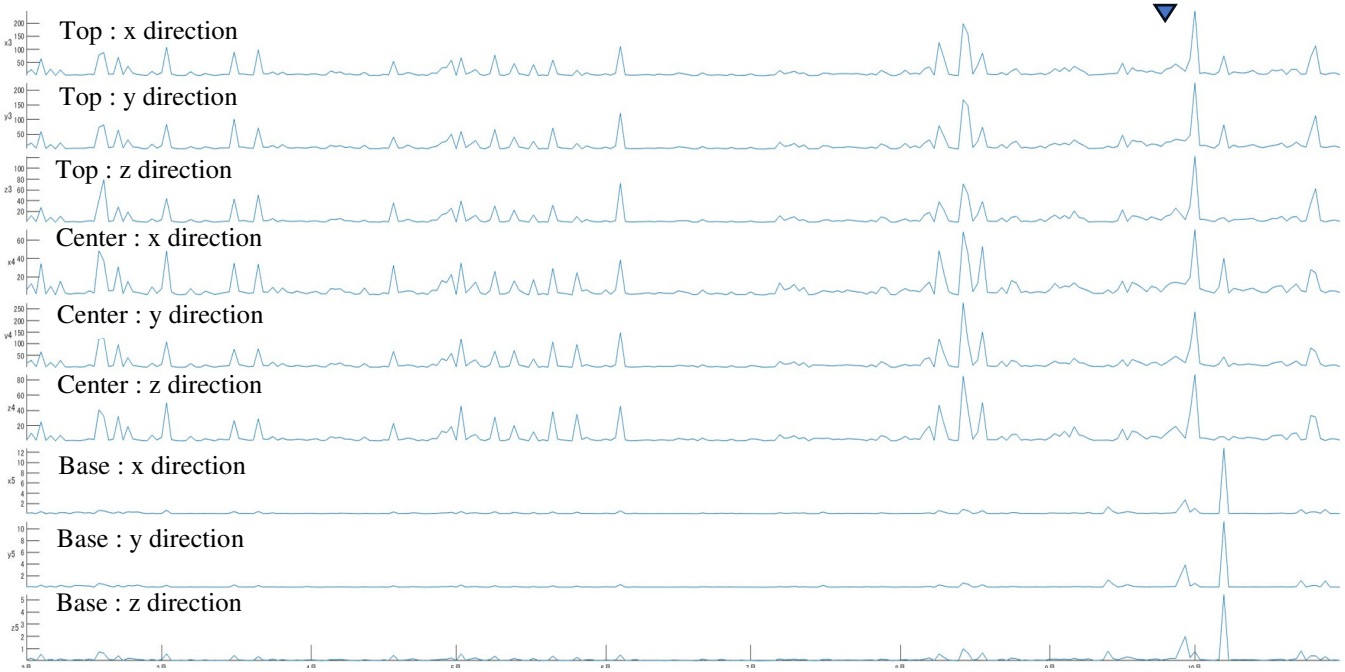
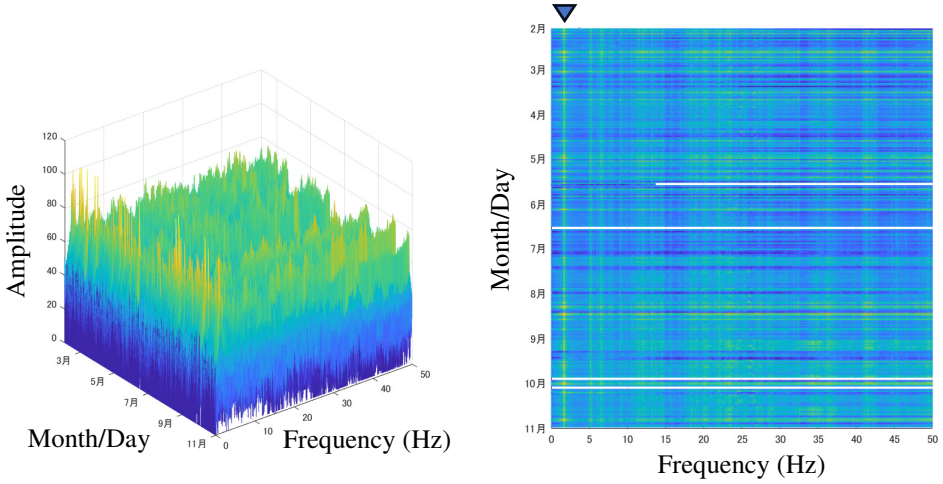
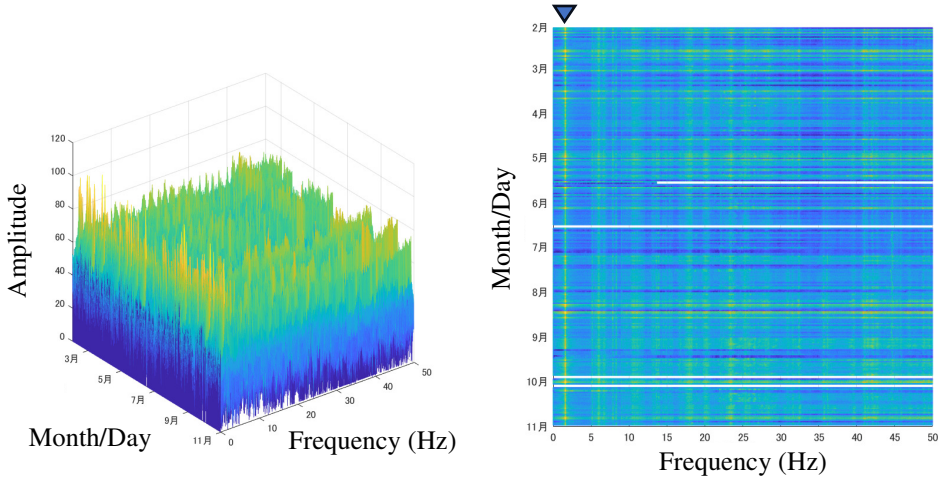


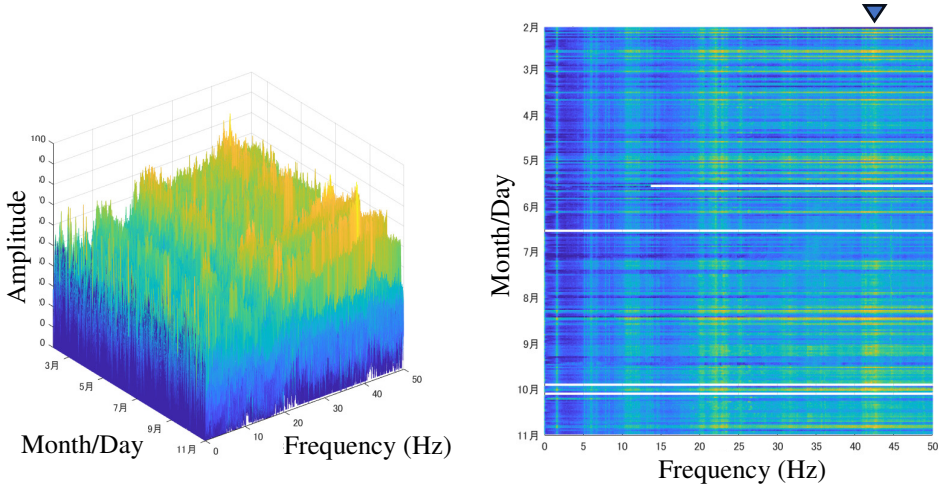
Figure 2. Maximum values measured by high-precision accelerometers during the measurement period.



(a) x direction

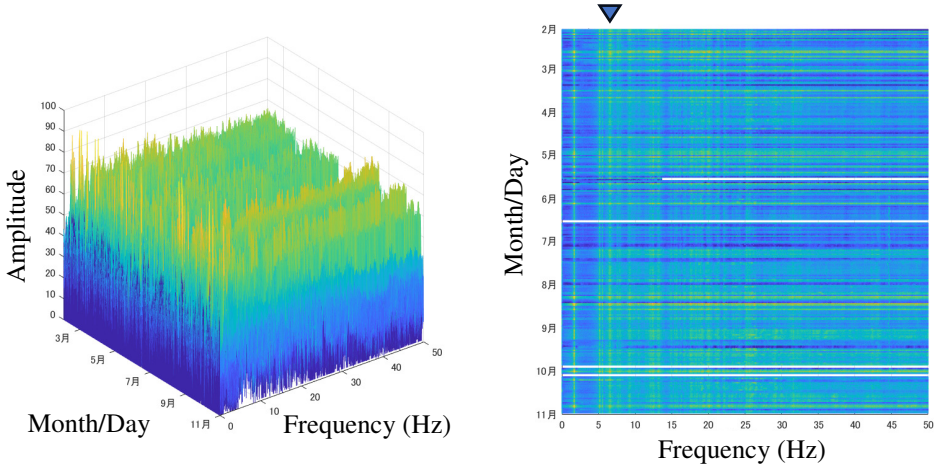


(b) y direction

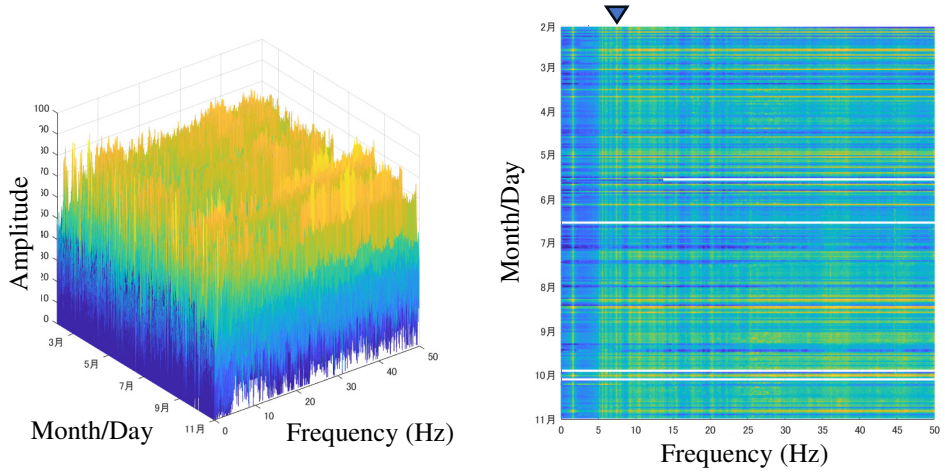


(c) z direction

Figure 3. Fourier spectra of measured data at the top of the tower obtained by high-precision accelerometer during the measurement period.



(a) x direction



(b) y direction

Figure 4. Fourier spectra of measured data at the top of the tower obtained by high-precision accelerometer during the measurement period.

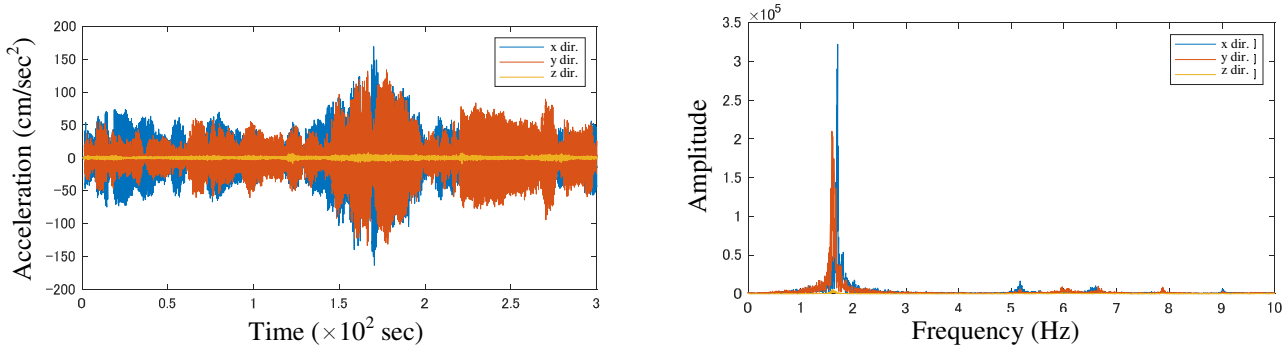


Figure 5. Fourier spectra of measured data at the top of the tower obtained by high-precision accelerometer during the measurement period.

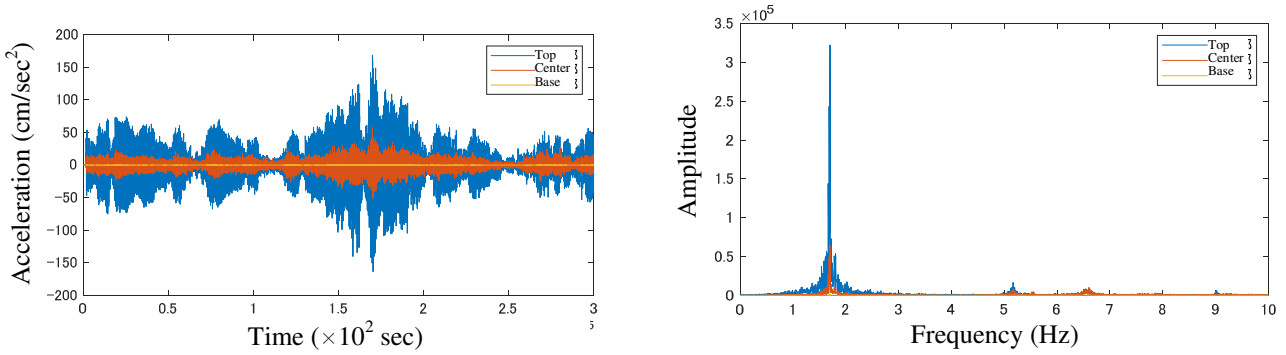


Figure 6. Fourier spectra of measured data obtained by high-precision accelerometer during the measurement period (x direction).

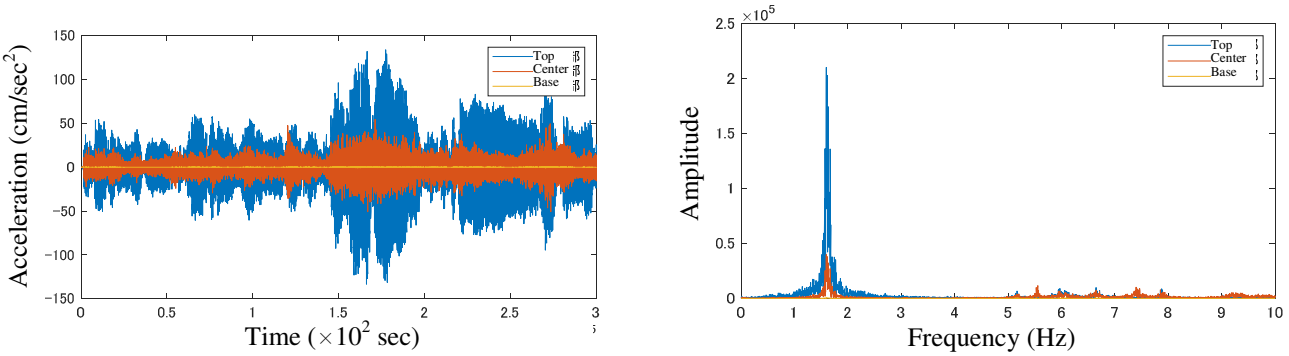


Figure 7. Fourier spectra of measured data obtained by high-precision accelerometer during the measurement period (y direction).

The steel tower structure analyzed in this study is subject to temperature effects, and its vibrations are induced by external wind forces. In the present measurement campaign, no sensors were installed to record temperature, wind speed, or wind direction. Therefore, meteorological data collected by the Japan Meteorological Agency (JMA), a national institution, were used as a reference for the investigation. The JMA operates approximately 50 meteorological observatories and weather stations across Japan, conducting meteorological observations and providing weather information. The observed parameters include temperature, humidity, atmospheric pressure, precipitation, wind direction, and wind speed, which are recorded continuously on a 24-hour basis, with data collected in real time. The accumulated data are archived and made publicly available via the web. The steel tower under investigation is located in Numazu City, Shizuoka Prefecture, and the nearest observation point is the Mishima Weather Station.

Figure 8 shows the average, maximum, and minimum temperatures recorded at this station during the measurement period. The maximum temperature reached 36.3°C, the minimum temperature was -3.5°C, and the average temperature varied gradually between 4°C and 30°C, remaining slightly above the normal seasonal range. No extreme temperatures that could significantly affect the structural performance of the steel tower were observed.

Figure 9 presents the average and maximum wind speeds recorded during the measurement period. From February to March, westerly seasonal winds prevailed, with an average wind speed of approximately 3.8 m/s. In April, the wind direction shifted to the south, and from May to August, southerly winds predominated, with wind speeds decreasing to around 3 m/s, the lowest values recorded throughout the year. In September, due to the influence of typhoons and other factors, easterly winds gradually increased, and in October, easterly winds became dominant, accompanied by a slight increase in wind speed. The Fourier spectra shown in Figures 3 and 4 indicate no temporal variation in the natural frequencies, suggesting that changes in wind speed and direction did not have a significant influence on the structural behavior.

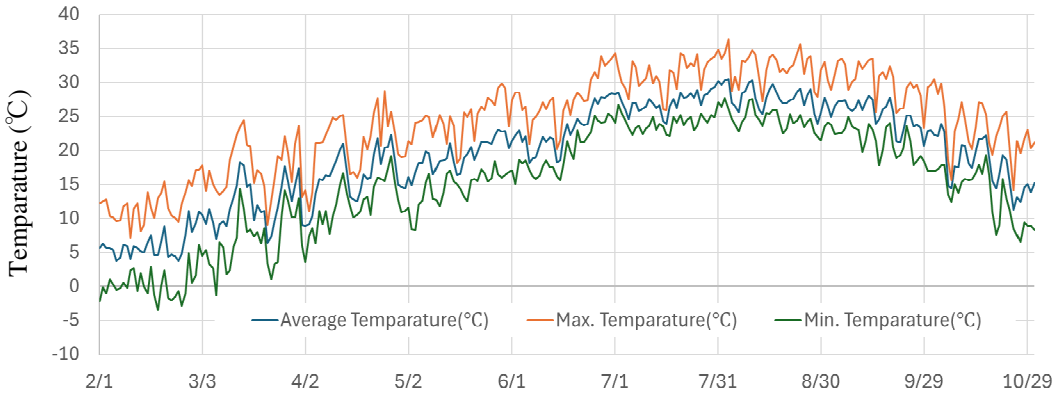


Figure 8. Average, maximum and minimum temperatures observed at JMA observation station.

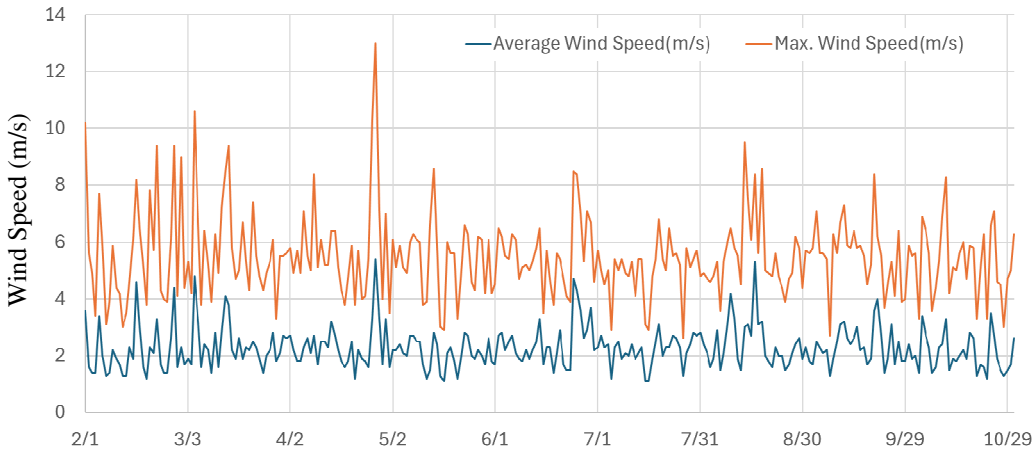


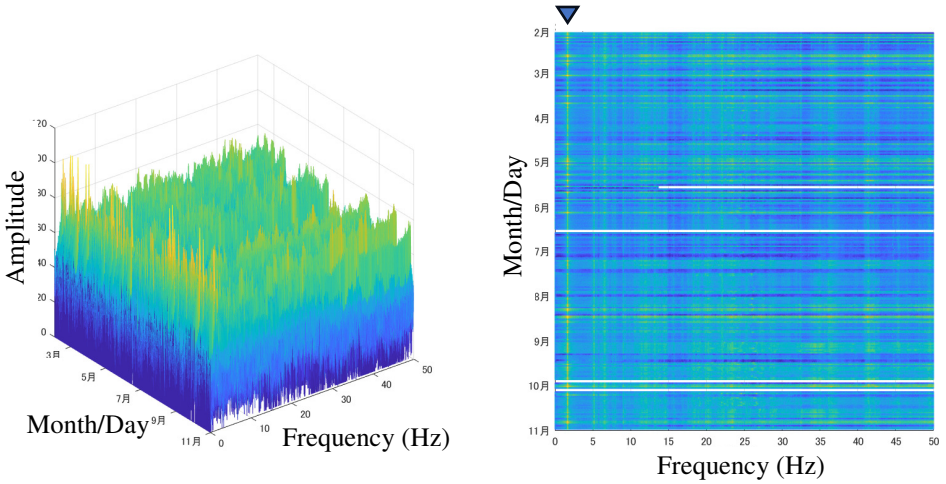
Figure 9. Average and maximum wind speeds observed at JMA observation station.

V. Comparison of High-precision Accelerometers and MEMS Accelerometers

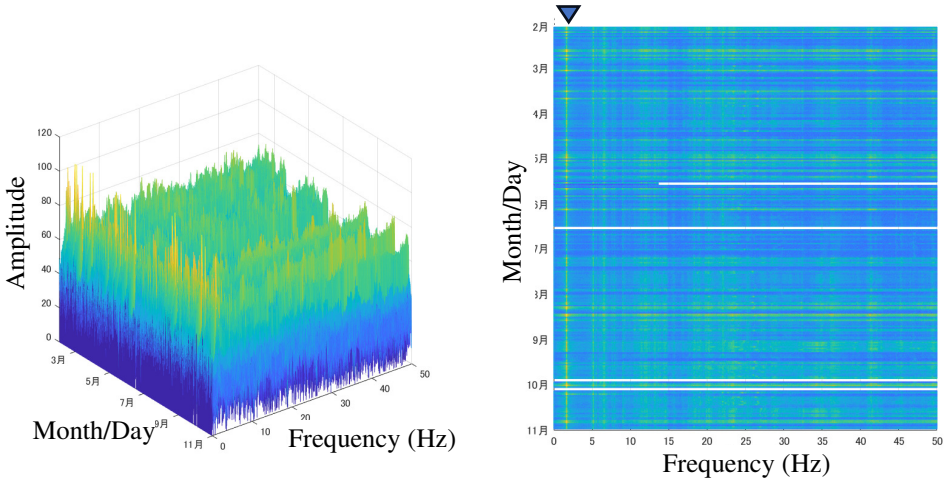
As mentioned above, accelerometers were used to measure the acceleration at the top, center and base of the tower, and both high-precision accelerometers and inexpensive MEMS accelerometers were installed in each part of the tower. Although it is desirable to use high-precision accelerometers for any analysis, in view of the widespread use of measurement systems, it is important to determine the extent to which analysis is possible with MEMS accelerometers which are not highly accurate, but very inexpensive. The following comparisons were therefore made, and the practicality of MEMS accelerometers was examined.

Figure 10 shows the Fourier spectra of the measured data (x-direction) at the top of the tower by the high-precision accelerometer and MEMS accelerometer during the measurement period. As shown in Table II, MEMS accelerometers have a lower noise density performance than high-precision accelerometers, making it difficult to accurately measure small accelerations. Therefore, when comparing the right diagrams in Figures 10(a) and (b), MEMS accelerometers gave a lower overall clarity of the diagram. However, the peaks indicated by ▼ were still captured in the same manner as those of the high-precision accelerometers, and it is possible to discriminate natural frequencies as an indicator of damage risk.

The day when the maximum acceleration of the measurement period occurred is indicated by ▼ in Figure 2. Figures 11 and 12 compare the time history waveforms of the data measured at the top of the tower by the high-precision accelerometer and the MEMS accelerometer at this time. Figures 11 and 12 show that the measurement results from the high-precision accelerometers and the MEMS accelerometers are almost identical, indicating that there are no problems with measurement at large amplitudes, which is important for discriminating natural frequencies as a damage risk indicator. Although the use of high-precision accelerometers is preferable for data analysis, it was confirmed that the use of inexpensive MEMS accelerometers is sufficient in actual practice.



(a) Measurement result by high-precision accelerometer(x direction)



(b) Measurement result by MEMS accelerometer(x direction)

Figure 10. Fourier spectra of measured data at the top of the tower obtained by high-precision and MEMS accelerometer (x direction).

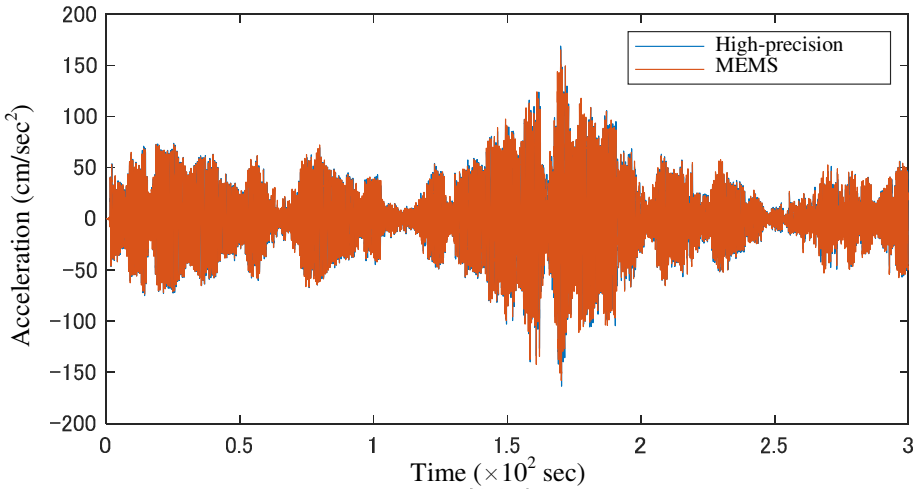


Figure 11. Comparison of time history waveforms of the data measured by high-precision and MEMS accelerometer (x direction).

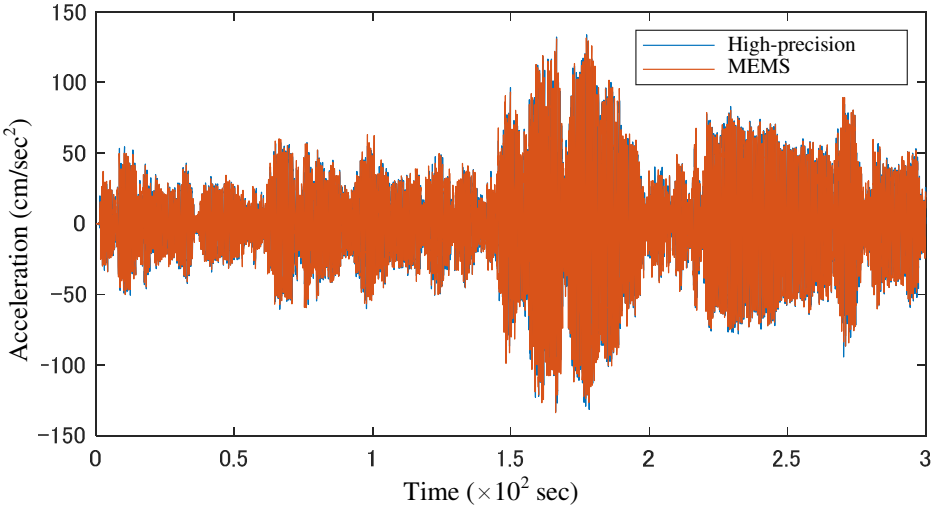


Figure 12. Comparison of time history waveforms of the data measured by high-precision and MEMS accelerometer (y direction).

VI. CONCLUSION

In the present study, sensors were installed on steel towers, and measurement data was collected and analyzed over a period of several months in order to obtain knowledge useful for monitoring the health of steel tower structures, extending their service life, and equalizing and facilitating reconstruction work. From the measurements and data analysis over a period of nine months, it was found that in the targeted steel tower structures, vibrations at the first-order natural frequencies were dominant, but the higher-order mode natural frequencies were also identified. Continuous monitoring of the natural frequencies of tower structures is considered effective as an indicator of damage risk. If a change in the damage risk indicator occurs, specifically if a trend towards lower natural frequencies is detected, it can be assumed that the rigidity of the structure has decreased, and this can lead to a higher priority for detailed inspection and planned reconstruction work. As a future issue, although the deterioration of the structural performance of the tower over time can be obtained through such fixed-point data measurement and analysis, it would be desirable to acquire data on events during typhoons and other strong winds and earthquakes, when the risk of tower collapse is high, together with video images to understand what events are occurring at that time. and to develop a measurement system to ensure their time synchronization.

ACKNOWLEDGMENT

This research was supported by the Ohata Zaidan, to which the author would like to express gratitude and was partially supported by JSPS KAKENHI Grant Number JP23K04342.

REFERENCES

[1] N. Kurata, “Research on Structural Health Monitoring of Steel Tower Structure Using Long-term Vibration Sensing,” The Fifteenth International Conference on Sensor Device Technologies and Applications (SENSORDEVICES 2024) IARIA, Nov. 2024, pp. 41-47, ISSN: 2308-3514, ISBN: 978-1-68558-208-1.

[2] Investigation and review committee regarding the tunnel ceiling collapse accident, “Report of the Investigation and Review Committee on the Tunnel Ceiling Collapse Accident,” Ministry of Land, Infrastructure, Transport and Tourism, http://www.mlit.go.jp/road/ir/ir-council/tunnel/pdf/130618_houkoku.pdf, 2013, in Japanese

[3] Trade and Industry Industrial Safety Group, “Current status of technical standards for steel towers and utility poles,” Ministry of Economy, Trade and Industry, https://www.meti.go.jp/shingikai/sankoshin/hoan_shohi/denryoku_anzen/tettou/pdf/001_04_00.pdf, 2019, in Japanese

[4] Working Group for the Investigation and Review of Damage Accidents of Steel Towers and Utility Poles Caused by Typhoon No. 15 of 2019, “Report of Working Group for the Investigation and Review of Damage Accidents of Steel Towers and Utility Poles Caused by Typhoon No. 15 of 2019,” https://www.meti.go.jp/shingikai/sankoshin/hoan_shohi/denryoku_anzen/tettou/pdf/20200121_report_01.pdf , 2020, in Japanese

[5] C. Boller, F. Chang, and Y. Fujino, “Encyclopedia of Structural Health Monitoring,” John Wiley & Sons, Sep. 2009, ISBN: 978-0-470-05822-0, DOI: 10.1002/9780470061626.

[6] Y. Liu and S. Nayak, “Structural Health Monitoring: State of the Art and Perspectives,” JOM 64, pp. 789–792, 2012, ISSN: 1543-1851, DOI: 10.1007/s11837-012-0370-9.

[7] Y. Bao et al., “The State of the Art of Data Science and Engineering in Structural Health Monitorng,” Engineering, vol. 5, Issue 2, pp. 234-242, 2019, ISSN: 2096-0026, DOI: 10.1016/j.eng.2018.11.027.

[8] P. C. Chang, A. Flatau, and S. C. Liu, “Review Paper: Health Monitoring of Civil Infrastructure,” Structural Health Monitoring, vol. 2, Issue 3, pp. 257-267, 2003, ISSN: 1475-9217, DOI: 10.1177/1475921703036169.

[9] J. P. Lynch, H. Sohn, and M. L. Wang, "Sensor Technologies for Civil Infrastructures," vol. 1 : Sensing Hardware and Data Collection Methods for Performance Assessment, ISBN: 978-0-08-102696-0, DOI: 10.1016/C2017-0-02662-9.

[10] C. R. Farrar and K. Worden, "Structural Health Monitoring: A Machine Learning Perspective," John Wiley & Sons, Jan. 2013, ISBN: 978-1-119-99433-6, DOI:10.1002/9781118443118.

[11] A. R. Al-Ali et al., "An IoT-Based Road Bridge Health Monitoring and Warning System," Sensors, vol. 24, pp. 469-491, 2024, ISSN: 1424-8220, DOI:10.3390/s24020469.