Data-Driven Energy Modeling of Machining Centers Through Automata Learning

Livia Lestingi, *Member, IEEE*, Nicla Frigerio, *Member, IEEE*, Marcello M. Bersani, Andrea Matta, *Senior Member, IEEE*, and Matteo Rossi

***Abstract*— The paper addresses the problem of estimating the energy consumed by production resources in manufacturing so that alternative process designs can be compared in terms of energy expenditure. In particular, the proposed methodology focuses on Computer Numerical Controlled (CNC) machining centers. Classical approaches to energy modeling require high expertise and large development effort since, for example, data acquisition is resource-specific and must be repeated frequently to avoid obsolescence. An automated and flexible data-driven methodology is designed in this work. A data-driven method is employed to learn a hybrid and stochastic model of a CNC machining center's energetic behavior. The learned model is used to provide offline energy consumption estimates of simulated part-programs before the actual execution of the cutting. Numerical results show the performance of the proposed method on a set of case studies. The methodology is also applied to a real industrial application, including data collected during machine production.**

***Note to Practitioners*—This article provides a flexible and autonomous data-driven approach to building models representing the energetic behavior of production resources, particularly CNC machining centers. The learned models can predict machine energy consumption while executing complex part-programs. The algorithm uses data that are commonly acquired by contemporary machine monitoring systems and does not require ad-hoc experimental tests for training. Specifically, it requires the spindle rotary speed signal, part load/unload signal, and spindle (or machine) power signal during the learning phase, whilst the estimation phase uses only the load/unload and spindle speed simulated signals.**

***Index Terms*— Energy modeling, machining, automata learning, data-driven modeling.**

I. Introduction

ENERGY consumption reduction is a clear need of contemporary manufacturing. Effective energy-efficient measures must be supported by proper models of the energy behavior of production resources, enabling the assessment and monitoring of their energy consumption. Indeed, estimating the energy consumed by manufacturing resources is crucial to evaluate alternative processes in terms of energy expenditure at the process design stage, where information might be very limited.

In particular, Computer Numerical Controlled (CNC) machining centers are the resources of interest of this work as considerable consumers of electrical energy [10], [12]. A single CNC machining center (hereinafter *machine*) might execute a large mix of chip removal tasks on numerous part types. The heterogeneity and complexity of machines make it difficult to create general and flexible models representing a large set of part-programs and to estimate their related energy consumption.

The automated discovery of energy models using limited information is an open research challenge. The ability to estimate machine energy consumption before the actual execution of a part-program enables the estimation of product footprint in terms of energy, the optimization of processes, the selection of alternative sequences and sets of tasks to obtain a certain workpiece, and the allocation of workpieces to alternative machines. Machine producers and users generally have a keen interest in techniques for developing flexible and, preferably, automated energy modeling techniques. In practice, the described ability is challenging due to the limited amount of information related to the actual execution of a certain machining operation; indeed, digital twins of the machine and the process are not common and, in their absence, the implementation of a sensing and monitoring system can be costly to measure, record, and maintain the information. The state-of-the-art on the topic lacks of solution for this case.

This paper addresses the problem of discovering a model of the machine's energetic behavior under limited information. The proposed data-driven methodology exploits an automata learning algorithm to obtain models of a machine executing complex operations (the part-program). The algorithm uses data acquired by the monitoring system of contemporary machines and does not require ad-hoc experimental tests for model training. Since machines are significant consumers of electrical energy, the paper focuses on electrical energy as the physical property under learning but can be extended to represent the consumption of other materials or utilities.

The rest of Section I describes related literature and details the paper's contribution. Section II provides preliminaries on Stochastic Hybrid Automata (SHA). Section III presents the

Manuscript received 29 November 2023; revised 11 March 2024 and 12 June 2024; accepted 15 July 2024. This article was recommended for publication by Associate Editor Y. Tong and Editor T. Nishi upon evaluation of the reviewers' comments. This work was supported in part by the European Union (EU) Horizon Europe Research and Innovation Program under Grant 101092021 (AutoTwin). *(Corresponding author: Livia Lestingi.)*

Livia Lestingi and Marcello M. Bersani are with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milan, Italy (e-mail: livia.lestingi@polimi.it).

Nicla Frigerio, Andrea Matta, and Matteo Rossi are with the Department of Mechanical Engineering, Politecnico di Milano, 20156 Milan, Italy.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2024.3430394.

Digital Object Identifier 10.1109/TASE.2024.3430394

problem statement and the application domain. Section IV details the proposed methodology. Numerical results are presented for a lab-scale application in Section V and for a real industrial case in Section VI. Section VII concludes.

A. Related Literature

Given the recent increase of interest in the topic, the literature includes systematic overviews on machine energy assessment and modeling, e.g., [26], [30], and [33]. The surveys discuss classical approaches to building machine models from prior knowledge of the cutting process, the machine, and the product. Such approaches can obtain analytical, empirical or hybrid models. Analytical models capture the energy consumption of specific machining tasks (e.g., turning, milling, drilling) as a known function of cutting parameters (spindle speed, depth of cut, feed rate) and of the workpiece and tool materials and geometry. Empirical models, despite being generally simpler than analytical approaches, require ad-hoc and extensive experimental tests to fit model parameters. For example, [12] modeled the machine-specific energy consumption as a function of the material removal rate. Hybrid models integrate measured data and theoretical knowledge of the cutting process [17], [20], [32] or combine nominal parameters, theoretical knowledge, and data retrieved from the machine controller during ad-hoc designed experiments [1], [9], [22]. Consequently, the characterization of specific machines and tasks can be effortful in terms of model development and tuning.

AI and Machine Learning (ML) techniques are also used to train machine energy models. For instance, [6], [13] use an Artificial Neural Network (ANN) trained over a set of ad-hoc experiments on milling tasks. Reference [31] trained a Neural Network (NN) to obtain a model of a lathe. Reference [19] incorporated transfer learning and Bayesian and Markov chain Monte Carlo calibration, then introduced a random forest regression as the base learner to train the prediction model. Reference [18] also exploits the random forest algorithm. For this group of works, the large number of physical variables to be acquired and the custom-designed experiments to obtain training data limit the applicability of the approaches. Also, the learned models are not amenable to formal verification, and they are not human-readable, being a black-box representation of machine behavior.

We note that the emulation of the part-program execution and/or the interpretation of the Numeric Control (NC) code executed by the machine are used in literature to understand machine operations at a given time and to obtain an accurate model. These approaches require a high customization effort to be applied to a specific machine, PLC, and part. NC program interpretation in [25] is coupled with empirical data in the NN training; a similar approach is proposed in [5], [7], and [29]. To assume that the acquisition system has access the NC program is the main barrier to applying these approaches in practice.

Finally, the main limitations of available approaches are the need for ad-hoc sets of experiments for algorithm and model training, the need for a significant amount of diverse signals and parameters to understand the energy consumption of machining tasks, the high level of expertise needed to build the model and the lack of flexibility. Moreover, while interpreting why an AI-based model produces a certain prediction becomes an increasingly pressing issue due to the potential economic impact, ML models are typically considered *opaque* rather than transparent [4]. In this regard, discovering an *explainable* model of machine behavior, such as an automaton, sheds light on the dynamics that lead to a specific energy prediction value.

B. Paper Contribution

This paper addresses the industrial need for flexible and automated methodologies to model the energy consumption of a machine while executing a certain sequence of tasks. Differently from state-of-the-art methods, this methodology uses data with limited diversity that can be acquired in the industrial practice with a limited effort. The proposed methodology uses a data-driven learning approach to create the model of the energy required by the machine to execute a part-program.

The methodology envisages two main phases: learning and usage. During learning, the algorithm uses the machine power signal acquired over time (i.e., the physical property under learning) and two signals representing machine behavior—the so-called *mined signals*—that are the spindle rotary speed and a signal indicating part load/unload events. During the usage phase, an energy consumption estimate is obtained through the learned model using only the mined signals as input, which are simulated without executing any cutting process.

Unlike other works in the literature, the proposed approach does not require specific knowledge of the machine, the part, tool geometry, materials, or the executed cutting task. The required information is limited to signals commonly acquired by contemporary machine monitoring systems, and training does not require ad-hoc experimental tests. Thus, the proposed approach overcomes specific shortcomings of existing approaches as it can be applied, for example, when the part mix is unknown or too broad to be exhaustively tested in the field or when information on the nature of the machining tasks is unavailable or too costly to acquire.

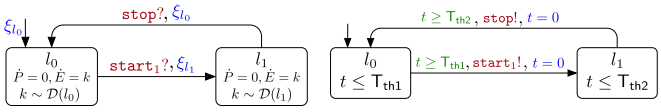
The idea behind the proposed approach is that a machine data-acquisition system includes signals that are related to the execution of a part-program and that such signals can be used to guide a model of the machine's physical behavior—i.e., machine power requests over time.

The learning algorithm called $\mathsf{L}^*_{\mathrm{SHA}}$ [16] is employed in the proposed approach. $\mathsf{L}^*_{\mathrm{SHA}}$ creates the machine energy predictive model, which relies on an automata-based formalism. The approach presented in this paper exploits $\mathsf{L}^*_{\mathrm{SHA}}$ within a broader methodology. Also, compared to [16], the algorithm has been enriched with a post-processing phase of the learned model to estimate probability distributions of the power request through Kernel Density Estimation (KDE) [21], [24], and with a trace-based simulation technique to calculate the energy estimate. In addition to the methodology, these extensions are also contributions presented in this paper.

With $\mathsf{L}^*_{\mathrm{SHA}}$, each predictor is realized through a SHA endowed with probability distributions modeling the machine power request for a certain machining task. Due to its stochastic nature, the learned SHA is amenable to statistical techniques to estimate the machine power request for a certain part-program, thus predicting machine energy consumption. Unlike ML models, human-interpretable SHA not only provide

TABLE I
LIST OF SYMBOLS

Symbol	Description
$l_i \in L$	SHA location named l_i
$\mathcal{F}(l_i)$	Flow condition labeling l_i
P, E	Power and energy consumption
k	Constant energy derivative
$\mathcal{D}(l_i)$	Prob. Distribution of k in l_i
t	Time spent in an SHA location
m	N. of spindle speed bins
`load`, `unload`, `stop`	Event labels on SHA edges
$\texttt{start}_j, j \in [1, m]$	
ξ_{l_i}	Update on edges incoming l_i



(a) Machine. (b) Controller.

Fig. 1. Example SHA modeling a machine/controller pair.

a specific energy estimation, but also lend themselves to wider-ranging evaluations, e.g., bottleneck analysis and strategy synthesis.

The proposed methodology is compared with a standard *baseline* estimation of the energy computed as the sample mean of the energy consumption of the machine during the training period (irrespective of speed variations). Other literature methods cannot be applied due to a lack of information available for the problem at hand (e.g., part and tool geometry, cutting parameters, material, NC part-program).

II. PRELIMINARIES

The models learned through $\mathsf{L}^*_{\text{SHA}}$ are SHA, a directed graph-based formalism enabling modeling the temporal evolution of complex systems' behavior (e.g., machines). Table I reports the symbols used in this paper for SHA. The graph's nodes (i.e., *locations*, which belong to set L) capture the states through which the system cycles. Fig. 1a shows an example of SHA with locations $L = \{l_0, l_1\}$: l_0 represents the initial idle state of the machine and l_1 models a machining task during which the machine requires energy.

SHA feature set W of real-valued variables whose time dynamics are constrained through sets of Ordinary Differential Equations (ODEs), called *flow conditions* [2], which model physical behaviors and belong to $\{\mathbb{R} \rightarrow \mathbb{R}^W\}$. Function $\mathcal{F}$ assigns the set of flow conditions to each location, constraining the behavior of real-valued variables of W while in such location. In Fig. 1a, set W contains variables P (resp. E) modeling the power request (resp. energy consumption) of a machine. Both P and E are functions of time t and an independent, randomly distributed term k, indicated as $P(t,k)$ and $E(t,k)$, respectively. For all locations $l_i \in L$, flow condition $\mathcal{F}(l_i)$ constrains the time derivative of P and E while in l_i, indicated as $\dot{P}(t,k)$ and $\dot{E}(t,k)$, with domain $\mathbb{R}_+ \times \mathbb{R}$. The SHA of Fig. 1a is such that $\mathcal{F}(l_i) = \langle \dot{P}(t,k) = 0, \dot{E}(t,k) = k \rangle$ holds for all locations $l_i \in \{l_0, l_1\}$, meaning that power request $P(t,k)$ is constant in time, while the consumed energy $E(t,k)$ is linear in time with slope k. SHA feature function $\mathcal{D}$ assigning a *probability distribution* to each location. For our analysis, for every location $l_i \in L$, $\mathcal{D}(l_i)$ characterizes the distribution of k (thus, $P(t,k)$ and $E(t,k)$) while in l_i (i.e., when the machine behaves as modeled by location l_i). Therefore, $P(t,k)$ and $E(t,k)$ behave as two stochastic processes.

Edges of SHA between two locations model the transition between two machine modes after the occurrence of a system event. Every edge is labeled with: a *guard condition* (in green), the *event* triggering the transition (in red), and an *update* (in blue). The guard condition is a logical expression on the variables in W: the expression must evaluate to true for the edge to be able to fire (e.g., the edge from l_0 to l_1 in Fig. 1b cannot fire if $t < \mathsf{T}_{\text{th1}}$ holds). The update is a set of instructions executed when the edge fires, computing new values for variables in W. Such values can either result from arithmetical expressions or by extracting a sample from a known probability distribution. In Fig. 1a, any edge entering a location $l_i \in L$ is labeled with update ξ_{l_i}, which assigns a sample from $\mathcal{D}(l_i)$ [11] to k. For example, when the SHA switches to l_0, update ξ_{l_0} is performed, and a sample from the distribution $\mathcal{D}(l_0)$ is extracted and assigned to k.

Multiple SHA form a network and synchronize through events by simultaneously performing a transition through edges with the same event label. Given event $\texttt{c} \in C$, an edge can be labeled either with `c`! or with `c`?. If an SHA in the network takes an edge labeled with `c`!, another automaton must take an edge labeled with the complementary label `c`?. For example, the SHA in Fig. 1a (i.e., the machine) switches from l_0 to l_1 when the controller fires event $\texttt{start}_1$ and from l_1 to l_0 when event `stop` occurs. Our analysis is limited to *input deterministic* SHA (i.e., no location has more than one outgoing edge with the same event label).

A *trace* is a sequence of events. For the network of Fig. 1, sequence $\texttt{start}_1$, `stop`, $\texttt{start}_1$, `stop` is a possible trace representing two working phases of the machine. The number of events is the *length* of a trace. A *timed trace* is a sequence of pairs (t_i, c_i), where i is an integer, $t_i \in \mathbb{R}_+$ is a timestamp, and c_i is the event occurring at time t_i. Sequence $(\mathsf{T}_{\text{th}_1}, \texttt{start}_1)$, $(\mathsf{T}_{\text{th}_1} + \mathsf{T}_{\text{th}_2}, \texttt{stop})$, $(2\mathsf{T}_{\text{th}_1} + \mathsf{T}_{\text{th}_2}, \texttt{start}_1)$, $(2\mathsf{T}_{\text{th}_1} + 2\mathsf{T}_{\text{th}_2}, \texttt{stop})$ is a possible timed trace for the network of Fig. 1. The *duration* of a timed trace is the last timestamp.

The controller features real-valued variable t to measure the duration of the machining task performed by the automaton in Fig. 1a while in *busy* and in *idle* modes. The ODEs describing the dynamics of t are $\dot{t} = 1$ in both locations, as t simply measures the time elapsing while the controller automaton is in l_0 and l_1. Every location l_i of an SHA can be endowed with an *invariant*, i.e., a condition over variables in W that must hold as long as the SHA is in l_i. In Fig. 1b, the combination of invariants of the form $t \leq \mathsf{T}_{\text{th}i}$, where $\mathsf{T}_{\text{th}i}$ is a constant value such that $i \in \{1, 2\}$ and $\mathsf{T}_{\text{th}i} \in \mathbb{Q}_+$ hold, and guards on outgoing edges of the form $t \geq \mathsf{T}_{\text{th}i}$ ensures that edges fire exactly when $t = \mathsf{T}_{\text{th}i}$ holds. As a result, the idle phase (corresponding to the controller's location l_0) lasts T_{th1} time units, and the working phase (corresponding to the controller's location l_1) lasts T_{th2} time units (at the onset of the system all variables are set to 0 unless differently specified).

$\mathsf{L}^*_{\text{SHA}}$ is applied to learn the following features of an SHA modeling a machine through signals collected from the field: 1) the set of locations L; 2) the set of edges, and in particular, for each edge, its starting and target locations, and the associated event label $\texttt{c} \in C$; 3) for each

location $l_i \in L$, flow conditions $\mathcal{F}(l_i)$, chosen out of a set of candidate functions; 4) for each location $l_i \in L$, probability distribution $\mathcal{D}(l_i)$.

Stochastic analysis of an SHA network can be carried out, for example, through the Uppaal verification tool [15], which extracts an arbitrarily large number of traces from the SHA network and collects the time evolution of its variables. In particular, by applying Monte Carlo-based simulations, it is possible to compute the expected value of distributions defined by means of functions `max` (maximum) and `min` (minimum) applied to stochastic processes in W. For example, given an upper bound τ and the automata of Fig. 1, formula $\mathbb{E}_\tau[\texttt{max}(E)]$ corresponds to the expected value of the maximum energy required by machine executions that last at most τ time units.

III. Assumptions and Application Domain

This work produces an SHA of the machine that represents machine power behavior in time while executing a certain part-program and is used to estimate the associated machine energy consumption. The proposed methodology requires:

- the signal related to the physical property under learning;
- the mined signals necessary for event mining, i.e., to identify the events in the traces.

The learned SHA reflects the correlation between mined signals and the physical property under learning. Machine behavior is represented in terms of locations and edges, and it is elicited from the analysis of the collected signals and the temporal relationships among them. Since the final objective is to represent machine energy expenditure, the physical property under learning is the power request. The modeling problem is tackled through partial knowledge of the machining task in execution, so we consider only two mined signals:

1) a signal indicating whenever a part-program is executed from which part load/unload events are mined;
2) the spindle rotary speed signal, since the spindle speed is known as one of the most significant factors affecting the energy consumption in machining.

The model learning phase is executed offline and exploits signals acquired during machine production—i.e., a set of signals (referred to as *dataset*) acquired during the execution of a set of workpieces. Prior knowledge about the executed part-programs is not required for the learning phase, and it is not used in the algorithm. Then, given the mined signals of a certain part-program, the learned model is used to represent the behavior of the machine and to provide an estimate of machine energy consumption using such mined signals to run a trace-based simulation. The estimation phase does not require the actual execution of the part-program on the machine.

A. Application Scenarios

To highlight the flexibility of the modeling approach, we leverage three application scenarios, each representing a typical use of industrial machines and with increasing complexity for energy estimation. The assumptions for each scenario are:

a. Scenario A (SA). *The machine is used to process high volumes of standard products.* Thus, the machine is assumed to be devoted to the machining of a single part type, as when installed in automated transfer lines.
b. Scenario B (SB). *The machine is used to process a family of products.* For example, machines are installed in production lines dedicated to a set of product types with similar features. For SB, we assume that the machine works on different part types and that n_B part-programs are executed by the machine, one for each part type.
c. Scenario C (SC). *The machine is used to produce highly customized products.* Machines are typically working as stand-alone solutions, executing part-programs shaped by client requests, where the production of individual workpieces can be entirely different. For SC, the number of part-programs executed by the machine is $n_C \gg n_B$, where each workpiece may represent a new part type.

Scenarios SA and SB are two lab-scale applications and scenario SC is a real industrial application. All scenarios make use of datasets acquired from real machines. In SA and SB, the machine is a vertical machining center (VGC1500) equipped with a maximum power of 18.5kW and a maximum spindle speed of 10,000rpm. Data are obtained through lab-scale experiments executing dry-cutting end-milling tasks on a steel workpiece. Additional details can be found in [17]. As for SC, a traveling column machining center for extremely heavy stock removal on workpieces (with a maximum payload of 5,000kg) with a maximum swing of 2100mm is considered. The machine is equipped with a horizontal electro-spindle (HSK100 45/70kW, maximum 10,000rpm, 5 linear axes), and all auxiliary systems (i.e., high-pressure coolant fluid system, cooling fluid filtering circuit, motor and axis chiller, chip extracting system, air extractor). This machining center is installed at a producer of hydraulic manifold blocks as a stand-alone solution in the machining department, and it is connected to an online monitoring system acquiring signals from the field.

In all cases, the machine monitoring system integrated with the PLC collected the *spindle power* signal—i.e., the physical quantity under learning—the *spindle rotary speed* signal and the *clamping pressure* signal related to the workpiece blocking system. The latter has been used to mine part load/unload events because high pressure indicates the workpiece is loaded. Fig. 2 represents an example of signal acquisition during the production of a workpiece in SC.

B. Domain Assumptions

The proposed methodology is developed and validated under specific assumptions related to the application domain.

We assume that the selected mined signals are significant for the estimation of machine energy, which is supported by theoretical knowledge of chip removal processes. Although many other cutting parameters (e.g., depth of cut, feed rate, tool geometry) are involved, their effect is not explicitly modeled. Such approximation is reasonable considering that spindle speed is a commonly monitored signal, whilst other information might need a significant effort to be extracted.

We assume that the production of a workpiece is independent of the sequence of previously produced workpieces and, in turn, that the energy consumption is not affected, either. To this end, the acquired signals are partitioned into portions capturing a single part-program (i.e., the corresponding trace respectively begins/ends with part load/unload events).

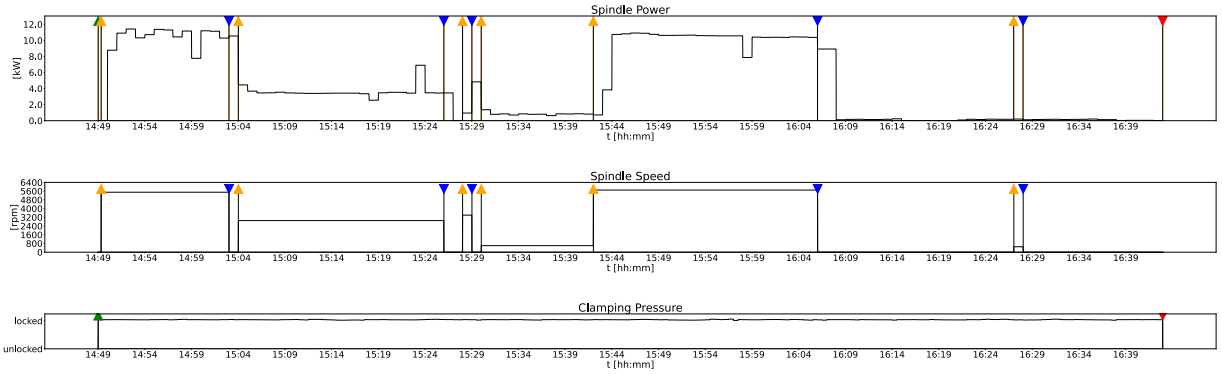


Fig. 2. Example of acquisition for the industrial case (*SC*). The three signals acquired for the part-program are represented over time. Mined events are marked *green* for part load, *red* for part unload, *yellow* for speed changes and *blue* for spindle reaching zero-speed (i.e., a stop).

We assume that the same acquisition frequency is used for clamping pressure, spindle speed, and spindle power so that signals are synchronous. Therefore, for each power measurement, $\mathsf{L}^*_{\text{SHA}}$ requires a speed and a pressure measurement with the same timestamp. Whenever acquisitions are not synchronous, the signal with the least frequent acquisitions must be used as a reference to batch other signals.

The SHA formalism assumes that there are finitely many locations representing the machine's behavior and events. In this work, the term *event* indicates a significant variation of a monitored signal, where a domain expert determines the conditions for significance. Events capture pressure and speed changes, which are possibly highly volatile due to sensor noises or intrinsic variability of sources. Signals are pre-processed to reduce the number of identified events.

We assume that the power request between two consecutive events—i.e., between two consecutive changes of spindle speed—is constant. In reality, as depicted in Fig. 2, the spindle power signal varies between consecutive events, showing that this assumption implies an approximation. Indeed, the power request depends on several cutting parameters that might vary even if the speed does not. Therefore, power request is modeled as the result of a stochastic process, whose distribution is assumed to be stationary in time but not known in advance.

IV. Proposed Methodology

The proposed methodology's workflow is represented in Fig. 3. All phases are performed automatically and do not require manual effort on the practitioner's side.

A. Data Collection and Processing

Given the data-driven nature of the learning algorithm $\mathsf{L}^*_{\text{SHA}}$, the data collection (phase **1** in Fig. 3) is necessary to acquire *field data*. A portion of the collected data is used for training and the remaining for validation. As the output of the data processing (phase **2** in Fig. 3), *timed traces* are extracted from field data. Thus, data processing operations include (1) the identification of events from mined signals (i.e., spindle speed and clamping pressure in our case) and (2) the creation of individual training traces by slicing signals.

Therefore, let us consider a mined signal $x(t)$ and let $t = t_0 t_1 \dots t_n$ be the sequence of timestamps at which an acquisition is performed. We indicate as $\mathcal{L} : \mathbb{R}_+ \to C \cup \{\bot\}$ the labeling function that, given a timestamp t_i, determines whether an event in set C has occurred at time t_i ($\mathcal{L}(t_i) = \bot$ holds if no event occurred at time t_i). Finally, data are sliced into timed traces as a trace is intended to represent the production of a single workpiece (i.e., from the loading of the workpiece to the following unloading).

1) Event Mining: The considered mined signals are the spindle speed and the clamping pressure. Let $\mathrm{p}(t_i) \in \mathbb{R}_+$ be the clamping pressure value measured at time t_i and $\mathrm{s}(t_i) \in \mathbb{R}_+$ be the spindle speed value measured at time t_i. The machining task starts only if the workpiece is loaded into the machine. Events `load` and `unload` represent the beginning and the end of part-program execution. The machine's hydraulic workpiece clamping system blocks the workpiece with a high-pressure value. Thus, signal $\mathrm{p}(t_i)$ is used to identify `load` and `unload` events for sudden clamping pressure variations (see Eq.1, where $\mathsf{p}_{\mathsf{min}}$ is the minimum pressure needed to block the workpiece).

Variations in spindle speed might result in a different power request. Thus, speed variations are events to be extracted from signal $\mathrm{s}(t_i)$. The spindle speed signal is discretized into a sequence of step-like signals (see Fig. 2) so that small speed variations are not taken into account, and its range, technologically limited to $[0, \mathsf{s}_{\mathsf{max}}]$, where $\mathsf{s}_{\mathsf{max}}$ is the nominal maximum speed of the machine's spindle, is divided into m bins of equal width $\delta_\mathrm{s} = \mathsf{s}_{\mathsf{max}}/m$ (where m is a configurable parameter). In the scenarios addressed in this paper, $\mathsf{s}_{\mathsf{max}} = 10,000$rpm and $m = 50$ according to expert knowledge. Events are identified whenever the spindle speed signal changes from bin to bin; thus, we label $\texttt{start}_j$ (with $j = 1 \dots m$) the event of the spindle reaching a speed within the j-th range, defined as interval $[(j-1)\delta_\mathrm{s}, j\delta_\mathrm{s})$. When the spindle speed is $\mathrm{s}(t_i) = 0$, the spindle stops (event `stop`).

Function $\mathcal{L}$ is fully given in Eq.1.

$$\mathcal{L}(t_i) = \begin{cases} \texttt{load}, & \text{if } i > 0 \wedge \mathrm{p}(t_i) \geq \mathsf{p}_{\mathsf{min}} \wedge \mathrm{p}(t_{i-1}) < \mathsf{p}_{\mathsf{min}} \\ \texttt{unload}, & \text{if } i > 0 \wedge \mathrm{p}(t_i) < \mathsf{p}_{\mathsf{min}} \wedge \mathrm{p}(t_{i-1}) \geq \mathsf{p}_{\mathsf{min}} \\ \texttt{stop}, & \text{if } i > 0 \wedge \mathrm{s}(t_i) = 0 \wedge \mathrm{s}(t_{i-1}) > 0 \\ \texttt{start}_j, & \text{if } i > 0 \wedge (j-1)\delta_\mathrm{s} \leq \mathrm{s}(t_i) < j\delta_\mathrm{s} \wedge \\ & (\mathrm{s}(t_{i-1}) \geq j\delta_\mathrm{s} \vee \mathrm{s}(t_{i-1}) < (j-1)\delta_\mathrm{s}) \\ \bot, & \text{otherwise} \end{cases} \tag{1}$$

For example, Fig. 2 marks with triangles the events identified by function $\mathcal{L}$ given the sample field data (e.g., $\mathcal{L}(14{:}49) = \texttt{load}$, $\mathcal{L}(15{:}04) = \texttt{start}_{11}$, and $\mathcal{L}(15{:}29) = \texttt{stop}$).

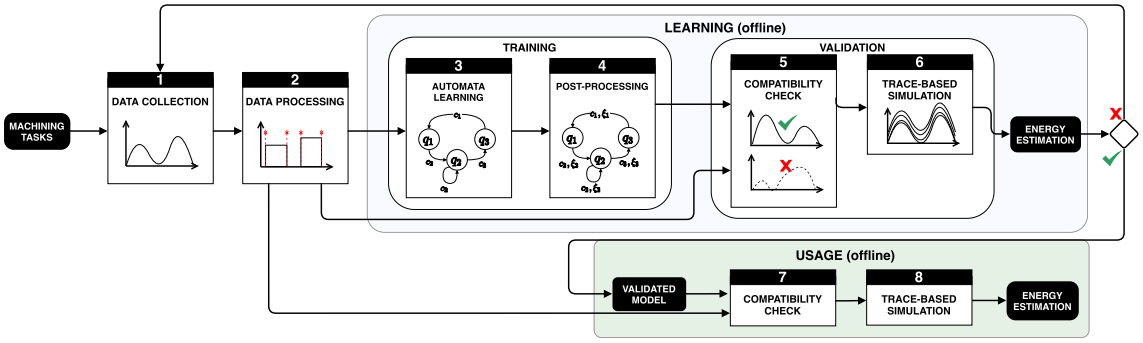


Fig. 3. Diagram representing the proposed methodology, split between two macro-phases: model learning and usage for prediction. Each phase is associated with a representation of its output and numbered according to the execution order.

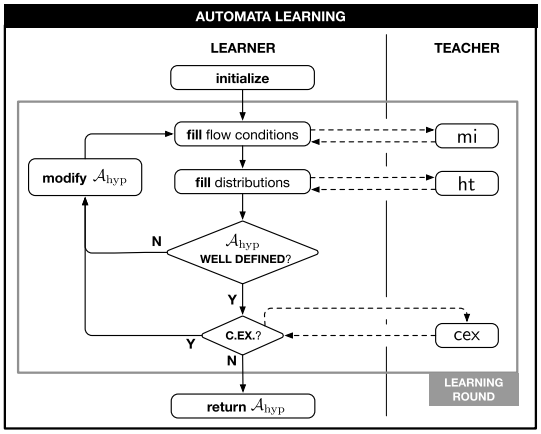
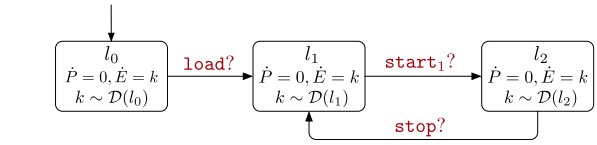
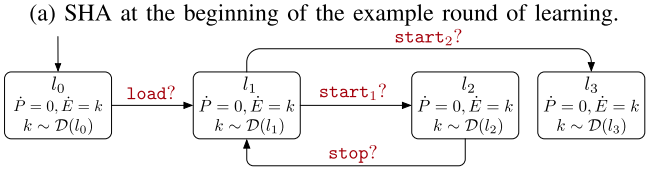


Fig. 4. $\mathsf{L}^*_{\text{SHA}}$'s workflow. Tasks are split between the learner and the teacher. Diamonds represent a bifurcation depending on a condition. Solid arrows connect two tasks executed sequentially, while dashed arrows represent a function invocation.



(a) SHA at the beginning of the example round of learning.



(b) SHA at the end of the example round of learning.

Fig. 5. Example SHA modeling a machine's power request and energy consumption. Each location is represented by its label, flow condition, and probability distribution. Notation $\dot{x}$ indicates the time derivative of a variable x.

2) Creation of Individual Traces: Sampled signals are sliced to capture the machining of a single workpiece. To this end, a simplified version of function $\mathcal{L}$ only identifies events `load` and `unload`. Timestamps at which such events are identified determine how signals are sliced before switching to the model training phase. Let $p \in \mathbb{N}$ be the number of identified events and $t' = t'_0 t'_2 \dots t'_{p-1}$ the sequence of timestamps at which an event is identified (i.e., $\mathcal{L}(t'_k) \neq \perp$ holds for all $k = 0, \dots, p-1$). The sequence of identified events $\mathcal{L}(t'_0) \dots \mathcal{L}(t'_{p-1})$ constitutes a trace of length p and duration t'_{p-1}.

B. Training

The training dataset is fed to the $\mathsf{L}^*_{\text{SHA}}$ automata learning algorithm (phase **3** in Fig. 3), followed by the post-processing (phase **4** in Fig. 3) to obtain the SHA modeling the machine power request in response to given events.

1) Automata Learning: $\mathsf{L}^*_{\text{SHA}}$, first introduced in [16], directly takes after the L^* algorithm developed by Angluin [3] for Deterministic Finite-state Automata learning and relies, at its core, on the interaction between a *learner* and a *teacher*. In L^*, the teacher is omniscient about the System Under Learning (SUL). However, omniscience is unfeasible for real-life black-box systems. Omniscience is sometimes approximated through testing (i.e., an input is sent to the real system to record the response behavior as output) [28], which, for machining centers, would require ad-hoc experiments and incur significant costs. For these reasons, in this work, the teacher is a data-storing and -processing component. The learner/teacher paradigm structures the software architecture behind the proposed methodology and does not have a physical counterpart. The physical component of the system consists of the machining center with sensors bridging the gap with the software component. Fig. 4 shows an overview of the algorithm executed during phase **3** of the methodology, while Fig. 5a and Fig. 5b are exploited as examples to illustrate the steps of the algorithm. In compliance with L^* (whose correctness is formally proved in [3]), the teacher and the learner provide different functions, which are all necessary (if invoked in the proper order) to achieve the correct result. The learner iteratively builds the hypothesis SHA, indicated as $\mathcal{A}_{\text{hyp}}$. The learner submits a query to determine a feature of $\mathcal{A}_{\text{hyp}}$ based on field data. The teacher is in charge of storing the accumulated knowledge about the SUL (i.e., field-collected data, specifically mined signals, traces, and the spindle power signal) and of processing such data to answer the learner's queries (e.g., applying hypothesis testing to identify the empirical distribution of power consumption during a specific machining operation). The accuracy of the teacher's answers depends on the techniques selected to implement the queries. $\mathcal{A}_{\text{hyp}}$ is initialized with a starting location (e.g., *init* in Fig. 5). The learning then proceeds in *rounds* where the learner refines $\mathcal{A}_{\text{hyp}}$ based on the teacher's answers to queries.

For each location $l_i \in L$ of $\mathcal{A}_{\text{hyp}}$, the learner must assign a flow condition (i.e., function $\mathcal{F}(l_i)$) and empirical probability distribution $X(l_i)$. To this end, the learner submits queries to the teacher, which exploits field data processed and labeled

as per Section IV-A to reply. The learner submits a mi query to determine the flow condition. The teacher identifies the best fit for the signals from a pre-determined set M of *candidate* functions constraining physical variables subject to uncertainty. To this end, the teacher exploits Derivative Dynamic Time Warping (DDTW) [14], which calculates the misalignment between two time series through warping along the x-axis and by comparing their first derivatives. In this application scenario, set M only contains tuple $\langle P(t,k) = k,\ E(t,k) = kt \rangle$ of stochastic processes with random parameter k, which is, thus, always returned as an answer to mi queries.

As per Section II and Section III-B, each location is characterized by the distribution of a random parameter—i.e., constant value k of variable P. The learner submits ht queries to the teacher to determine the distribution of k after the occurrence of events in a specific trace. Being power request stochastic, two portions of the power request signal, although unidentical, may capture the same behavioral state of the machine (i.e., be represented by the same location of the automaton under learning). For each field-collected population $K \subset \mathbb{R}_+$ of average spindle power demand, the learner submits a ht query to the teacher. For each previously identified population $K' \subset \mathbb{R}_+$, the latter performs a two-sample Kolmogorov-Smirnov test [23] to determine whether K and K' derived from the same distribution. If this is the case, K and K' are merged, and the resulting empirical distribution is returned as the answer to the ht query. Otherwise, K constitutes a *new* population, and the teacher returns its empirical distribution as the answer to the ht query. For example, let us assume that at the beginning of a learning round, the hypothesis SHA is as in Fig. 5a, and the learner submits a ht query to determine the empirical distribution underlying the mean power samples associated with trace $tr_{\text{ex}} = \langle \texttt{load}, \texttt{start}_1, \texttt{stop}, \texttt{start}_2 \rangle$. If the trace identifies a new machine's behavior, the ht query returns a different empirical distribution than the one assigned to location l_2.

Once all locations of $\mathcal{A}_{\text{hyp}}$ are labeled with a flow condition and an empirical distribution, the learner checks whether $\mathcal{A}_{\text{hyp}}$ is *well-defined*. If $\mathcal{A}_{\text{hyp}}$ is not well-defined, it is not amenable to simulation and cannot thus be returned by $\mathsf{L}^*_{\text{SHA}}$. The conditions for well-definedness are: 1) all edges connect existing locations ($\mathcal{A}_{\text{hyp}}$ is *closed*); 2) all edge targets are deterministic as per Section II ($\mathcal{A}_{\text{hyp}}$ is *consistent*). If either of the two conditions is unmet, the learner modifies $\mathcal{A}_{\text{hyp}}$. If $\mathcal{A}_{\text{hyp}}$ is not closed, the learner adds a new location. For example, after identifying the new empirical distribution for trace tr_{ex}, adding the location labeled as $busy_2$ in Fig. 5b makes the SHA closed. If $\mathcal{A}_{\text{hyp}}$ is not consistent, the learner splits a location into two new ones to solve the non-determinism. After modifying $\mathcal{A}_{\text{hyp}}$, new mi and ht queries must be submitted.

As the final step of the round, it is necessary to check whether $\mathcal{A}_{\text{hyp}}$ correctly captures all traces known by the teacher. To this end, the learner asks the teacher for a *counterexample* (shortened as *c.ex.* in Fig. 4) through a cex query. A counterexample is a trace of which the teacher stores observations but is not compliant with $\mathcal{A}_{\text{hyp}}$. If a counterexample exists, a new round of learning is necessary for the learner to properly modify $\mathcal{A}_{\text{hyp}}$ (i.e., a new location and/or new edges are added to the SHA). If the teacher finds no counterexample, $\mathsf{L}^*_{\text{SHA}}$ terminates and returns $\mathcal{A}_{\text{hyp}}$.

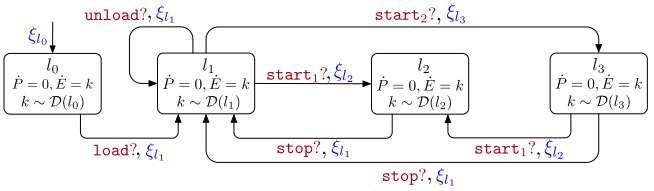
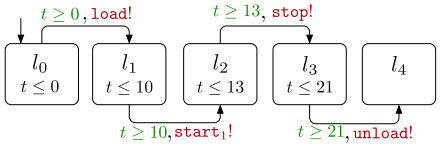
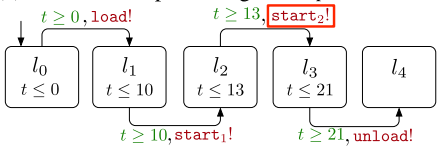


Fig. 6. Example post-processed SHA modeling machine energetic behavior. Updates are in *blue*.



(a) Controller reproducing a *compatible* trace.



(b) Controller reproducing a *incompatible* trace (the highlighted event causes the incompatibility).

Fig. 7. Controllers reproducing traces with the machine SHA in Fig. 5b. Each location has a name label and an invariant.

2) Post-Processing: Post-processing (phase **4** in Fig. 3) makes the learned SHA amenable to simulation. Specifically, edges of the learned SHA are extended with updates while a distribution estimate is assigned to each location. An example of post-processed SHA is shown in Fig. 6.

The SHA learned through $\mathsf{L}^*_{\text{SHA}}$ features, for every location $l_i \in L$, an empirical distribution identified by ht queries. $\mathsf{L}^*_{\text{SHA}}$ has no constraints on the specific shape of distribution functions; therefore, empirical functions identified by ht queries may converge to any arbitrary distribution.

Let $X(l_i) \subseteq \mathbb{R}_+$ be the sample set underlying the empirical distribution assigned to $l_i \in L$. The Gaussian Kernel Density Estimation (KDE) method [21], [24] is applied to $X(l_i)$, for each $l_i \in L$, to estimate $\mathcal{D}(l_i)$. KDE is a non-parametric method to estimate the probability density function of a random variable based on kernels as weights that do not imply any prior distribution. Gaussian KDE uses the normal kernel function. KDE requires the proper selection of a kernel smoothing parameter, called *bandwidth*, which is calculated through Silverman's approximation method [27]. For all locations $l_i \in L$ of the learned SHA, the empirical distribution is replaced with the kernel density estimator of $\mathcal{D}(l_i)$.

C. Validation

The objective of the validation phase is to assess the accuracy of a machine's energy estimate obtained from its learned SHA model. If the accuracy is deemed sufficient by the domain expert, then the SHA model can be released and thus become the predictor of the machine's energy expenditure during the execution of a possible machining trace (i.e., the *usage* macro-phase in Fig. 3). Otherwise, the SHA model is not accurate enough, and new training traces should be collected (loop in Fig. 3) to improve its accuracy. The validation dataset is used to validate the SHA resulting from post-processing.

The validation phase consists of two distinct activities, the compatibility check (phase **5** in Fig. 3) and the trace-driven simulation (phase **6** in Fig. 3), both relying on the notion of *trace compatibility*.

Verifying whether a trace is compatible with the machine SHA entails the generation of a *controller* SHA to compose a SHA network. The controller SHA mimics the *timed* sequence of events of the trace under analysis. The trace determines the structure of the controller SHA, which, thus, does not require learning through $\mathsf{L}^*_{\text{SHA}}$. In all controller SHA, $t \in W$ is a real-valued variable that grows uniformly with time (i.e., $\dot{t} = 1$ holds in all locations). An edge labeled with $\subset!$ with $\subset \in C$ cannot fire (thus, time cannot flow) unless another SHA in the network has an edge enabled with label $\subset?$. If all edges of the controller fire successfully, the machine SHA captures the trace under analysis—hence, the trace is *compatible* with the learned SHA. For instance, let us assume that the validation dataset contains two traces, $\langle \texttt{load}, \texttt{start}_1, \texttt{stop}, \texttt{unload} \rangle$ and $\langle \texttt{load}, \texttt{start}_1, \texttt{start}_2, \texttt{unload} \rangle$ with events at timestamps $\langle 10, 13, 21 \rangle$. Fig. 7 shows the two controller SHA mimicking the first and second trace, respectively, whose compatibility with the machine SHA in Fig. 6 needs to be verified. The controller in Fig. 7a mimics a trace compatible with the machine SHA in Fig. 6, while the trace mimicked by the SHA in Fig. 7b is incompatible because the edge from l_2 to l_3 cannot fire.

The compatibility check filters the validation dataset to identify traces that are compatible with the SHA under analysis. Trace-driven simulation is then executed for compatible traces only, as the machine's SHA does not model the machine's energetic behavior in response to sequences of events defining incompatible traces. If the number of compatible traces is found to be insufficient, the training and validation datasets should be re-calibrated (if possible, based on data availability) to iterate the learning and obtain a SHA with a better compatibility metric. To perform the compatibility check, the machine SHA is paired with the controller SHA mimicking the timed trace under analysis, and the resulting SHA network is simulated through the Uppaal tool. The tool generates a number of runs of the system in response to the events of the trace under analysis (fired by the controller SHA). Whenever an edge incoming location $l_i \in L$ fires, update ξ_{l_i} draws a sample of k from estimated $\mathcal{D}(l_i)$ through an acceptance/rejection algorithm [8]. Uppaal estimates the expected energy $\mathbb{E}_\tau[\texttt{max}(E)]$ consumed by the machine to execute the trace over the simulated time τ, which is equal to the duration of the timed trace. Finally, the estimated energy consumption is compared against field data to assess the learned model's accuracy.

V. Numerical Results

The methodology is applied to scenarios *SA* and *SB* (see Section III-A) to evaluate the accuracy of the energy estimate by varying the number of traces used in $\mathsf{L}^*_{\text{SHA}}$ training. Results obtained by applying the methodology to *SC* (i.e., the real industrial use case) are described separately. Experiments on all three scenarios have been performed on a machine running Ubuntu 22.04 with 64GB of memory and 4 cores.[1] Trace-based simulation is performed using Uppaal v.4.1.24.

[1] $\mathsf{L}^*_{\text{SHA}}$ implementation available at github.com/LesLivia/lsha

TABLE II
Part-Programs (Task Number as in [17])

Part type	Sequence of tasks
i	26-14-16-04-26-02
ii	18-05-25-24-10-12
iii	04-09-04-26-20-11
iv	26-14-16-04-26-02-18-24-18-04-19-06
v	18-05-25-24-10-12-27-06-20-10-21-26
vi	04-09-04-26-20-11-18-05-25-24-10-12

TABLE III
Summary of Experiments

ID	Scenario	Part mix	Part type	Trace length
$1-6$	A	Known	Single	$p = 14$ or $p = 26$
$7-9$	B	Known	Multiple	$p = 14$ or $p = 26$
10	C	Unknown	Multiple	$p \in [6, 21]$

TABLE IV
SA and *SB*: Example of Nominal Spindle Speeds ([rpm]) for a Generated Part-Program (Part Type i)

Tasks	26	14	16	04	26	02
Nominal speed	2500	1600	3200	2500	2500	3200

A. Description of Experiments

We designed a set of experiments to evaluate the performance of the proposed methodology in scenarios of increasing complexity, as described in Section III-A. Numerical results have been obtained for the following 9 problem instances (IDs from 1 to 9). A summary is in Table III:

- Six instances of *SA* are created starting from literature data. 27 machining tasks have been executed to acquire the spindle power with dedicated measures, as in Table VI of [17]. Spindle speed and clamping pressure signals have been generated according to the design information provided in [17]. The six part-programs in Table II, each one intended for the production of a single workpiece, have been randomly designed from [17]: part types i, ii, iii require a sequence of 6 tasks (thus, with trace length $p = 14$), and part types iv, v, vi require 12 tasks (thus, with trace length $p = 26$). Tasks can be repeated to execute a certain part type (e.g., task 26 in type i) and can be shared between part types (e.g., type i and iii). The part mix includes a single part type: IDs from 1 to 6 refer to part types i to vi, respectively. Training and validation datasets have been generated by concatenating the available signals and adding white noise.
- Three instances of *SB* are generated following the approach described for *SA*, except that each instance includes a mix of $n_B = 3$ part types: ID=7 corresponds to types i, ii, iii, ID=8 to iv, v, vi, and ID=9 to i, iv, vi.

A training dataset is used for model learning, and the proposed methodology is applied to problem instances by increasing the number N_{tr} of training traces (Section V-B). We consider $\mathsf{N}_{\text{tr}} = 3, 10, 20$ for each produced part type.

A validation dataset is considered for each of the 9 problem instances so that the learned SHA are evaluated. In more detail:

- *SA* instances (ID1–ID6) are validated over $\mathsf{N}_{\text{val}} = 30$ traces for, on average, 11.1h of acquisition per instance;
- *SB* instances (ID7–ID9) are validated over $\mathsf{N}_{\text{val}} = 90$ traces (i.e., 30 traces for each involved part type)

TABLE V

$\mathsf{L}^*_{\text{SHA}}$ RUNNING TIME ([s]) BROKEN DOWN BY THE INSTANCE NUMBER (ID), PRODUCED PART MIX, NUMBER OF LOCATIONS ($|L|$), AND EDGES ($|\mathcal{E}|$) OF THE LEARNED SHA, AND NUMBER OF TRAINING TRACES N_{tr} WITH RELATED ACQUISITION TIME ([h])

ID	Part Mix	$\lvert L\rvert$	$\lvert\mathcal{E}\rvert$	N_{tr}	Acquisition Time [h]	Run Time [s]
		8	11	3	1.00	8.84
1	i	8	11	10	3.33	16.63
		8	11	20	6.65	18.40
		12	12	3	0.46	4.67
2	ii	12	12	10	1.53	8.35
		12	12	20	3.06	10.27
		11	12	3	0.93	7.97
3	iii	11	12	10	3.09	15.58
		11	12	20	6.18	19.94
		12	16	3	1.77	46.37
4	iv	12	16	10	5.91	52.44
		12	16	20	11.82	67.66
		17	19	3	0.95	32.29
5	v	17	19	10	3.16	48.87
		17	19	20	6.31	58.24
		23	24	3	1.55	58.03
6	vi	23	24	10	5.18	81.93
		23	24	20	10.35	99.78
		20	25	3×3	4.32	49.28
7	i, ii, iii	20	25	10×3	14.41	76.59
		20	25	20×3	28.82	146.21
		48	54	3×3	2.38	247.34
8	iv, v, vi	48	54	10×3	7.95	354.32
		48	54	20×3	15.89	659.59
		30	36	3×3	4.27	146.08
9	i, iv, vi	30	36	10×3	14.24	204.43
		30	36	20×3	28.48	385.04
	Unknown	15	34	5	5.48	568.05
10	Unknown	30	60	7	8.29	1216.49
	Unknown	42	83	9	12.78	4119.48
	Unknown	88	145	15	15.58	5991.6
	Unknown	139	167	20	18.47	48252.0

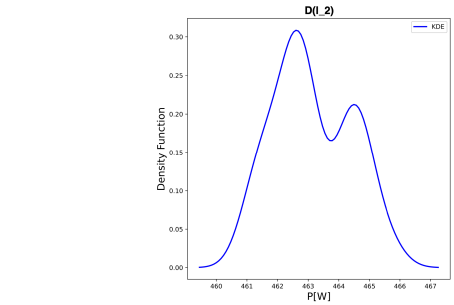
corresponding, on average, to 36.6h of acquisition per instance.

The number of traces compatible with the learned SHA is provided in Section V-C. As a measure of accuracy for the learned SHA, we exploit the mean relative error MRE_{est} (Section V-D). Let $E_{j,\text{est}}$ be the energy consumption *estimated* through the proposed methodology for validation trace j (with $j = 1 \ldots \mathsf{N}_{\text{val}}$). Let $E_{j,\text{ref}}$ be the reference energy consumption of trace j computed from the spindle power signal. We define the relative percentage error $\text{RE}_{j,\text{est}}$ for validation trace j as follows: $\text{RE}_{j,\text{est}} = \frac{|E_{j,\text{ref}} - E_{j,\text{est}}|}{E_{j,\text{ref}}} \cdot 100$. Then, the mean relative percentage error MRE_{est} is computed over the N_{val} traces.

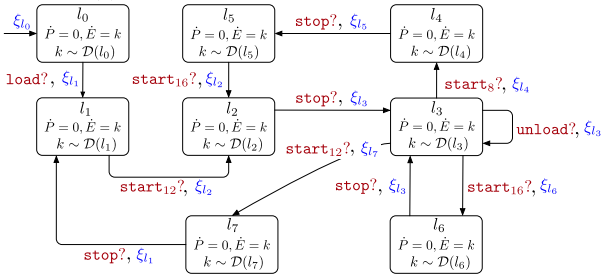
In addition, the proposed methodology is compared with the *baseline* estimation of the energy (Section V-D). Similarly to estimate $E_{j,\text{est}}$, the relative percentage error $\text{RE}_{j,\text{BL}}$ and its mean MRE_{BL} are computed.

B. Training Results

The size of the training dataset is proportional to N_{tr} and to the execution time of the part-program (i.e., to the time needed to acquire a single trace). The acquisition time is reported in Table V for each part type $i - vi$. For instances of SA, the average part-program execution time to obtain a single part is 1331s (22.19 minutes). Thus, the training dataset for a single instance of SA includes, on average, 1.1h, 3.7h, and 7.4h of data for, respectively, $\mathsf{N}_{\text{tr}} = 3, 10, 20$. Similarly, an instance of SB includes, on average, 3.7h, 12.2h, and 24.4h of data for, respectively, $\mathsf{N}_{\text{tr}} = 9, 30, 60$.



(a) Estimated P distribution in location l_2.



(b) Learned SHA.

Fig. 8. Output of $\mathsf{L}^*_{\text{SHA}}$ for SA, part type i, $\mathsf{N}_{\text{tr}} = 10$ training traces.

In all experiments, the running time of $\mathsf{L}^*_{\text{SHA}}$ (i.e., the computation time required by the algorithm to perform the learning) increases with the volume of the training dataset and the complexity of the learned SHA.

Concerning the SHA complexity, the number of locations and edges of the learned SHA (reported in Table V) indicates how the SHA complexity varies between different experiments. As Table V shows, the value of N_{tr} does not impact the number of locations and edges of the learned SHA because the training dataset for SA and SB always includes at least one trace for each product type. More precisely, for each SA instance, all traces obtained from the corresponding mined signals and fed as input to $\mathsf{L}^*_{\text{SHA}}$ feature the same sequence of events, as the produced workpieces belong to the same part type. Also, each instance of SB includes three part types, but at least one trace for each part type appears in the training dataset. Therefore, all learned SHA capture such traces, irrespective of N_{tr}.

Figure 8b shows an example of learned SHA, obtained for part type i with $\mathsf{N}_{\text{tr}} = 10$. Machining tasks constituting the part-program of part type i (see Table II) feature the spindle moving at three different nominal speeds (Table IV). Accordingly, the data processing of the speed signal captures three events $\texttt{start}_8$, $\texttt{start}_{12}$, and $\texttt{start}_{16}$ in addition to `stop`, `load` and `unload` events, which are trivially captured in all traces. The learned SHA has seven locations (in addition to the default l_0 location). Based on the estimated $\mathcal{D}(l_i)$, we can differentiate among locations l_2, l_4, l_6, and l_7—where the cutting process is in execution and the spindle requires non-negligible power—and locations l_1, l_3, and l_5—where the spindle stops and does not require any power. Locations l_1, l_3, and l_5 are reached after either `load`, `unload`, or `stop` events, as expected. Fig. 8a shows an example of a density

TABLE VI
ESTIMATION ERRORS BROKEN DOWN BY INSTANCE ID AND THE NUMBER OF TRACES N_{tr} USED FOR $\mathsf{L}^*_{\mathsf{SHA}}$ LEARNING

ID	N_{tr}	$\mathrm{MRE}_{\mathrm{est}}$	$\mathrm{MRE}_{\mathrm{BL}}$	ID	N_{tr}	$\mathrm{MRE}_{\mathrm{est}}$	$\mathrm{MRE}_{\mathrm{BL}}$
	3	0.918%	0.410%		3	1.240%	0.507%
1	10	0.592%	0.361%	6	10	1.198%	0.487%
	20	0.558%	0.349%		20	1.029%	0.167%
	3	0.720%	0.747%		3×3	4.482%	13.408%
2	10	0.691%	0.482%	7	10×3	4.255%	11.747%
	20	0.100%	0.109%		20×3	3.764%	9.370%
	3	0.566%	0.274%		3×3	3.292%	14.011%
3	10	0.475%	0.264%	8	10×3	2.839%	12.363%
	20	0.402%	0.246%		20×3	2.332%	12.060%
	3	1.393%	0.425%		3×3	3.787%	9.422%
4	10	1.041%	0.390%	9	10×3	3.304%	8.745%
	20	1.034%	0.388%		20×3	2.354%	8.478%
	3	1.319%	0.319%		5	37.67%	96.84%
5	10	1.208%	0.195%		7	26.75%	89.84%
	20	1.165%	0.120%	10	9	22.52%	84.05%
					15	22.04%	86.12%
					20	20.81%	86.71%

function estimated using the KDE method, specifically $\mathcal{D}(l_2)$. Although there are 6 machining tasks, the learned SHA has 4 locations modeling active machine's modes (l_2, l_7, l_6, and l_4), while the remaining locations model the machine's idle modes. Indeed, as shown in Table II, task nr. 26 is repeated twice, and location q_1 captures both repetitions. Also, task nr. 16 is not considered statistically different from task nr. 26; thus, $\mathsf{L}^*_{\mathsf{SHA}}$ groups them into a single location. Furthermore, locations modeling machining tasks (l_2, l_4, l_6, and l_7) only have outgoing edges labeled with the `stop` event. Similarly, locations modeling idle states (l_1, l_3, and l_5) only have outgoing edges with event labels $\texttt{start}_i$, with $i \in \{8, 12, 16\}$. As a consequence, if a simulation of the SHA of Fig. 8b is run, it is impossible to observe two consecutive `stop` events or two consecutive $\texttt{start}_i$ events, as this never occurs in the input field data.

C. Compatibility of Validation Traces

Perfect compatibility is achieved in all instances of SA as a consequence of domain assumptions (SA assumes the production of a single part type). Similarly, perfect compatibility is achieved in all instances of SB. Even if the part mix includes $n_B = 3$ part types, it has been assumed that all part types are experienced during the training phase. Therefore, all validation traces for ID1-ID9 are compatible with the learned SHA.

D. Validation Results and Energy Estimate

Each learned SHA is paired with a controller SHA mimicking a compatible trace, and a pair is defined for every trace in the validation dataset. Each SHA pair is imported into Uppaal to generate $\mathsf{N}_{\mathsf{val}}$ runs through simulation.

As discussed in Section I-A, existing techniques are not directly comparable to our methodology due to more restrictive sets of working assumptions (i.e., others are not applicable to these scenarios). Therefore, we exploit a sample mean as the baseline for the energy consumption metric. However, we recall that while the baseline is a simple numeric value, the human-interpretable learned models (amenable to several purposes other than energy estimation) represent the methodology's core result. Table VI reports the estimation errors obtained while comparing the SHA-based estimates of the energy consumption with the actual energy consumption as per the validation dataset. Obtained estimates yield, in the worst case, $\mathrm{MRE}_{\mathrm{est}} = 1.393\%$ and $\mathrm{MRE}_{\mathrm{est}} = 4.482\%$, respectively. Errors increase with trace length and with the product mix, specifically:

- SA: The estimation obtained through the learned SHA is slightly less accurate than the baseline (i.e., $\mathrm{MRE}_{\mathrm{est}} > \mathrm{MRE}_{\mathrm{BL}}$) in 16 cases out of 18 due to approximations introduced to fit distributions, which do not come into play with the baseline. On the other hand, the learned model (including such distributions) remains more informative than the simple point estimate while keeping a high degree of accuracy ($\mathrm{MRE}_{\mathrm{est}} < 1.4\%$ holds in all cases).
- SB: The learned SHA is more accurate than the baseline in all instances (i.e., $\mathrm{MRE}_{\mathrm{est}} < \mathrm{MRE}_{\mathrm{BL}}$). The learned SHA can capture the differences among the three produced part types using the spindle speed signals, whereas the baseline estimate only considers an average part type.

The baseline model is highly accurate for simple instances, but its performance worsens as problem complexity increases. The proposed methodology, instead, performs satisfactorily with more complex problems thanks to its ability to recognize tasks.

VI. REAL INDUSTRIAL CASE

This section describes the industrial application exploited to validate our methodology (i.e., scenario SC and instance ID10). Recall that the part mix is unknown, and each product is highly customized. Therefore, the trace mined from each acquisition differs from the others, and experiencing all part types at training time is unfeasible. The training and validation datasets have been collected from the production environment over the course of three weeks and contain 114 different traces of length $p \in [6, 21]$ (i.e., each trace includes 6 to 21 events). Timed traces last on average 1.44h, which contributes to the complexity of the addressed industrial case.

We present the results obtained by training $\mathsf{L}^*_{\mathsf{SHA}}$ with $\mathsf{N}_{\mathsf{tr}} = 5, 7, 9, 15, 20$ traces—corresponding to, approximately, 5.5h, 8h, 13h, 15.5h, and 18.5h of acquisition, respectively. Note that the limited number of traces actually represents a significant acquisition effort for the application field. Training results are reported in Table V (ID = 10). The run time needed for algorithm training increases from 9.4 minutes (568 s) to 13.4 hours (48252 s) as N_{tr} increases. Compared to SA and SB, the run time required to obtain the learned SHA for SC is significantly larger and implies a trade-off between the computation effort and the size of the training dataset.

Differently from SA and SB, the learned SHA locations and edges increase. Since the sequence of events changes for each training trace of SC—capturing a possibly new part type that is being produced—the structure of the learned SHA can vary significantly as the size of the training dataset increases.

Figure 9 shows the original and estimated power signals obtained with, for example, the SHA learned for SC with $\mathsf{N}_{\mathsf{tr}} = 9$ by simulating the trace mined from the signals shown in Fig. 2. As expected, the results collected in Table VI (ID = 10) show that errors decrease as the number of training traces increases. The improvement is more evident in SC than in SA and SB because each new trace might incorporate significant

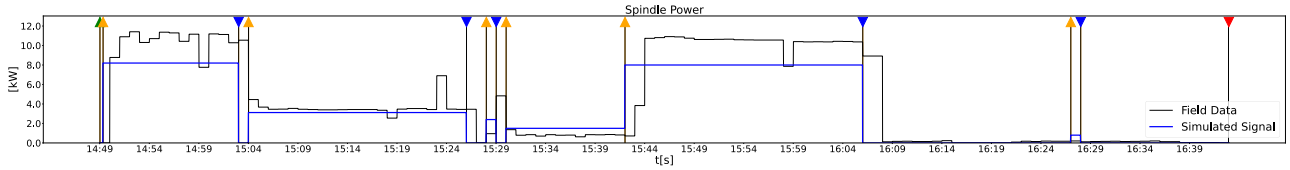


Fig. 9. Example of a sampled signal of the spindle power for *SC* (the same shown in Fig. 2) and power signal generated through Uppaal by simulating the behavior of the learned SHA. Events are marked as in Fig. 2.

new knowledge. The MRE_{est} improves from 37.67% to 20.81% with 15 additional traces. Despite errors being larger compared to *SA* and *SB*, the advantage of using the learned SHA compared to the baseline is evident, with an error gap (calculated as $\text{MRE}_{\text{BL}} - \text{MRE}_{\text{est}}$) of approximately 60%.

Numerical results for *SC* are obtained using $\mathsf{N}_{\text{val}} = 94$ traces, all different from one another and unseen during training. Therefore, a trace for *SC* may capture a different part-program, with a new sequence of tasks and/or never-before-seen tasks. As the number of traces in the training dataset increases, the probability of facing never-before-seen part types or tasks decreases. When a trace is not compatible with the learned SHA, partial compatibility is checked by generating a controller that mimics its longest compatible *prefix*.

In the validation dataset, 12, 27, 32, 49, and 56 traces are—at least partially—compatible with the SHA learned with $\mathsf{N}_{\text{tr}} = 5, 7, 9, 15, 20$, respectively. The average number of compatible events grows from 31% with $\mathsf{N}_{\text{tr}} = 5$ to 53% with $\mathsf{N}_{\text{tr}} = 20$ of the original trace length. *SC* displays a lower compatibility ratio than *SA* and *SB* due to the degree of customization of the involved part types, which yields a very diverse part mix. Nevertheless, the compatibility ratio increases with the value of N_{tr}. Moreover, should a desired trace be found to be incompatible, it is possible to add the trace to the training dataset and to update the SHA by repeating the training (thus guaranteeing its compatibility in the newly learned SHA).

VII. Conclusion

The presented methodology allows practitioners to reliably estimate the energy consumption of machining tasks when the part mix cannot be exhaustively tested during model training and when information about the nature of the machining tasks is unavailable or costly to obtain. The methodology can be fully automated in practice, thus entailing minor manual effort on the practitioner's side, and intrinsically open to iterative self-enhancements, e.g., by iteratively adding new traces to the training dataset. Obtained results show that the proposed method is fast and that it achieves very high accuracy in estimating machine energy consumption when the part mix is limited ($\text{MRE}_{\text{est}} \leq 1.393\%$ for *SA* and $\text{MRE}_{\text{est}} \leq 4.482\%$ for *SB*). The potential impact of the proposed approach has been investigated over a highly variable scenario resulting in $\text{MRE}_{\text{est}} = 22.52\%$ with only 9 part-programs used for training. The capability of capturing never-before-seen part-programs is shown in scenario *SC*, where the learned $\mathsf{L}^*_{\text{SHA}}$ can process new, yet at least partially compatible, traces.

The methodology can be extended to include other mined signals, such as the machining tool. In the future, we will investigate the impact of information on the estimate's accuracy. In addition, traces giving rise to behaviors that differ significantly with respect to a defined baseline might point to unexpected behaviors in machines; therefore, the proposed approach can become the foundation of a smart monitoring component. Future research efforts will focus on the estimation of never-before-seen sequences of events, currently labeled as *incompatible* with the learned model. This extension is paramount and might require the estimation of transition probabilities to pass from one machine state to another. Finally, the ability to handle general flow conditions constraining the machine power signature over time can be a significant extension. To this end, theoretical knowledge of the cutting process will be useful to guide the proposed approach.

Acknowledgment

The authors would like to thank Prof. Li and his research group for the additional details on their experiments in [17].

References

[1] P. Albertelli, A. Keshari, and A. Matta, "Energy oriented multi cutting parameter optimization in face milling," *J. Cleaner Prod.*, vol. 137, pp. 1602–1618, Nov. 2016.

[2] R. Alur et al., "The algorithmic analysis of hybrid systems," *Theor. Comput. Sci.*, vol. 138, no. 1, pp. 3–34, Feb. 1995.

[3] D. Angluin, "Learning regular sets from queries and counterexamples," *Inf. Comput.*, vol. 75, no. 2, pp. 87–106, Nov. 1987.

[4] A. B. Arrieta et al., "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, Jun. 2020.

[5] M. Brillinger, M. Wuwer, M. Abdul Hadi, and F. Haas, "Energy prediction for CNC machining with machine learning," *CIRP J. Manuf. Sci. Technol.*, vol. 35, pp. 715–723, Nov. 2021.

[6] M. A. Rodriguez-Cabal, A. Gonzalo, and S. Rudas, "Prediction of energy consumption in the leadwell V-40 it CNC machining center through artificial neural networks," *J. Appl. Eng. Sci.*, vol. 20, no. 1, pp. 145–149, 2022.

[7] J. Cao, X. Xia, L. Wang, Z. Zhang, and X. Liu, "A novel CNC milling energy consumption prediction method based on program parsing and parallel neural network," *Sustainability*, vol. 13, no. 24, p. 13918, Dec. 2021.

[8] G. Casella, C. P. Robert, and M. T. Wells, "Generalized accept-reject sampling schemes," in *Proc. LNMS*, 2004, pp. 342–347.

[9] K. Cheng, G. Zhao, W. Wang, and Y. Liu, "An estimation methodology of energy consumption for the intelligent CNC machining using STEP-NC," *Int. J. Adv. Manuf. Technol.*, vol. 123, nos. 1–2, pp. 627–644, Nov. 2022.

[10] J. B. Dahmus and T. Gutowski, "An environmental analysis of machining," in *Proc. Manuf. Eng. Mater. Handling Eng. (ASME)*, 2004, pp. 643–652.

[11] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "Uppaal SMC tutorial," *Int. J. Softw. Tools Technol. Transf.*, vol. 17, no. 4, pp. 397–415, Aug. 2015.

[12] T. G. Gutowski, M. S. Branham, J. B. Dahmus, A. J. Jones, A. Thiriez, and D. P. Sekulic, "Thermodynamic analysis of resources used in manufacturing processes," *Environ. Sci. Technol.*, vol. 43, no. 5, pp. 1584–1590, Mar. 2009.

[13] G. Kant and K. S. Sangwan, "Predictive modelling for energy consumption in machining using artificial neural network," *Proc. CIRP*, vol. 37, pp. 205–210, Jan. 2015.

[14] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proc. Intl. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2001, pp. 1–11.

[15] K. G. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *Int. J. Softw. Tools Technol. Transf.*, vol. 1, nos. 1–2, pp. 134–152, Dec. 1997.

[16] L. Lestingi, M. M. Bersani, and M. Rossi, "Model-driven development of service robot applications dealing with uncertain human behavior," *IEEE Intell. Syst.*, vol. 37, no. 6, pp. 48–56, Nov. 2022.

[17] C. Li, S. Wu, Q. Yi, X. Zhao, and L. Cui, "A cutting parameter energy-saving optimization method considering tool wear for multi-feature parts batch processing," *Int. J. Adv. Manuf. Technol.*, vol. 121, nos. 7–8, pp. 4941–4960, Aug. 2022.

[18] J. Liu, M. Li, and Y. Zheng, "A random forest regression-based method for predicting energy consumption of turning operations on CNC machine tools," in *Proc. ICEMCE*, Oct. 2023, pp. 650–654.

[19] F. Lu, G. Zhou, Y. Liu, and C. Zhang, "Ensemble transfer learning for cutting energy consumption prediction of aviation parts towards green manufacturing," *J. Cleaner Prod.*, vol. 331, Jan. 2022, Art. no. 129920.

[20] P. T. Mativenga and M. F. Rajemi, "Calculation of optimum cutting parameters based on minimum energy footprint," *CIRP Ann.*, vol. 60, no. 1, pp. 149–152, 2011.

[21] E. Parzen, "On choosing an estimate of the spectral density function of a stationary time series," *Ann. Math. Statist.*, vol. 28, no. 4, pp. 921–932, Dec. 1957.

[22] S. S. Pawar, T. C. Bera, and K. S. Sangwan, "Modelling of spindle energy consumption in CNC milling," *Proc. CIRP*, vol. 105, pp. 192–197, Jan. 2022.

[23] J. W. Pratt and J. D. Gibbons, "Kolmogorov–Smirnov two-sample tests," in *Concepts Nonparametric Theory*. New York, NY, USA: Springer, 1981, pp. 318–344.

[24] M. Rosenblatt, "Remarks on a multivariate transformation," *Ann. Math. Statist.*, vol. 23, no. 3, pp. 470–472, Sep. 1952.

[25] N. Shen, Y. Cao, J. Li, K. Zhu, and C. Zhao, "A practical energy consumption prediction method for CNC machine tools: Cases of its implementation," *Int. J. Adv. Manuf. Technol.*, vol. 99, nos. 9–12, pp. 2915–2927, Dec. 2018.

[26] N. Sihag and K. S. Sangwan, "A systematic literature review on machine tool energy consumption," *J. Cleaner Prod.*, vol. 275, Dec. 2020, Art. no. 123125.

[27] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Evanston, IL, USA: Routledge, 2018.

[28] B. Steffen, F. Howar, and M. Merten, "Introduction to active automata learning from a practical perspective," in *Proc. SFM*. Cham, Switzerland: Springer, 2011, pp. 256–296.

[29] R. Ströbel, Y. Probst, S. Deucker, and J. Fleischer, "Time series prediction for energy consumption of computer numerical control axes using hybrid machine learning models," *Machines*, vol. 11, no. 11, p. 1015, Nov. 2023.

[30] Y. Zhang, "Review of recent advances on energy efficiency of machine tools for sustainability," *Proc. Inst. Mech. Eng. B, J. Eng. Manuf.*, vol. 229, no. 12, pp. 2095–2108, Dec. 2015.

[31] G. Zhao, C. Hou, J. Qiao, and X. Cheng, "Energy consumption characteristics evaluation method in turning," *Adv. Mech. Eng.*, vol. 8, no. 11, Nov. 2016, Art. no. 168781401668073.

[32] X. Zhao, C. Li, Y. Tang, and Y. Lv, "An integrated decision-making method of flexible process plan and cutting parameter considering dynamic machining resources," *IEEE Trans. Autom. Sci. Eng.*, early access, Sep. 27, 2023, doi: 10.1109/TASE.2023.3315546.

[33] L. Zhou, J. Li, F. Li, Q. Meng, J. Li, and X. Xu, "Energy consumption model and energy efficiency of machine tools: A comprehensive literature review," *J. Cleaner Prod.*, vol. 112, pp. 3721–3734, Jan. 2016.

Livia Lestingi (Member, IEEE) received the Ph.D. degree in information technology from Politecnico di Milano in 2023. She is currently a Post-Doctoral Researcher with Politecnico di Milano. Her research interests include the analysis of human–robot interaction through formal methods and formal modeling techniques of human behavior.

Nicla Frigerio (Member, IEEE) is currently an Assistant Professor of Manufacturing and Production Systems with the Department of Mechanical Engineering, Politecnico di Milano, Milan, Italy. Her research interests include design and control of production systems and sustainable manufacturing automation.

Marcello M. Bersani is currently an Associate Professor with Politecnico di Milano, Milan, Italy. His research interests include formal methods, and temporal logic and verification.

Andrea Matta (Senior Member, IEEE) is currently a Full Professor of Manufacturing with the Department of Mechanical Engineering, Politecnico di Milano, Milan, Italy. His research interests include analysis, design, and management of manufacturing and health care systems, digital twin discovering and generation, stochastic processes, control systems, process mining, simulation-optimization, and sustainable automation.

Matteo Rossi is currently an Associate Professor with Politecnico di Milano, Milan, Italy. His research interests include formal methods for safety-critical and real-time systems, architectures for real-time distributed systems, and transportation systems both from the point of view of their design, and of their application in urban mobility scenarios.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement