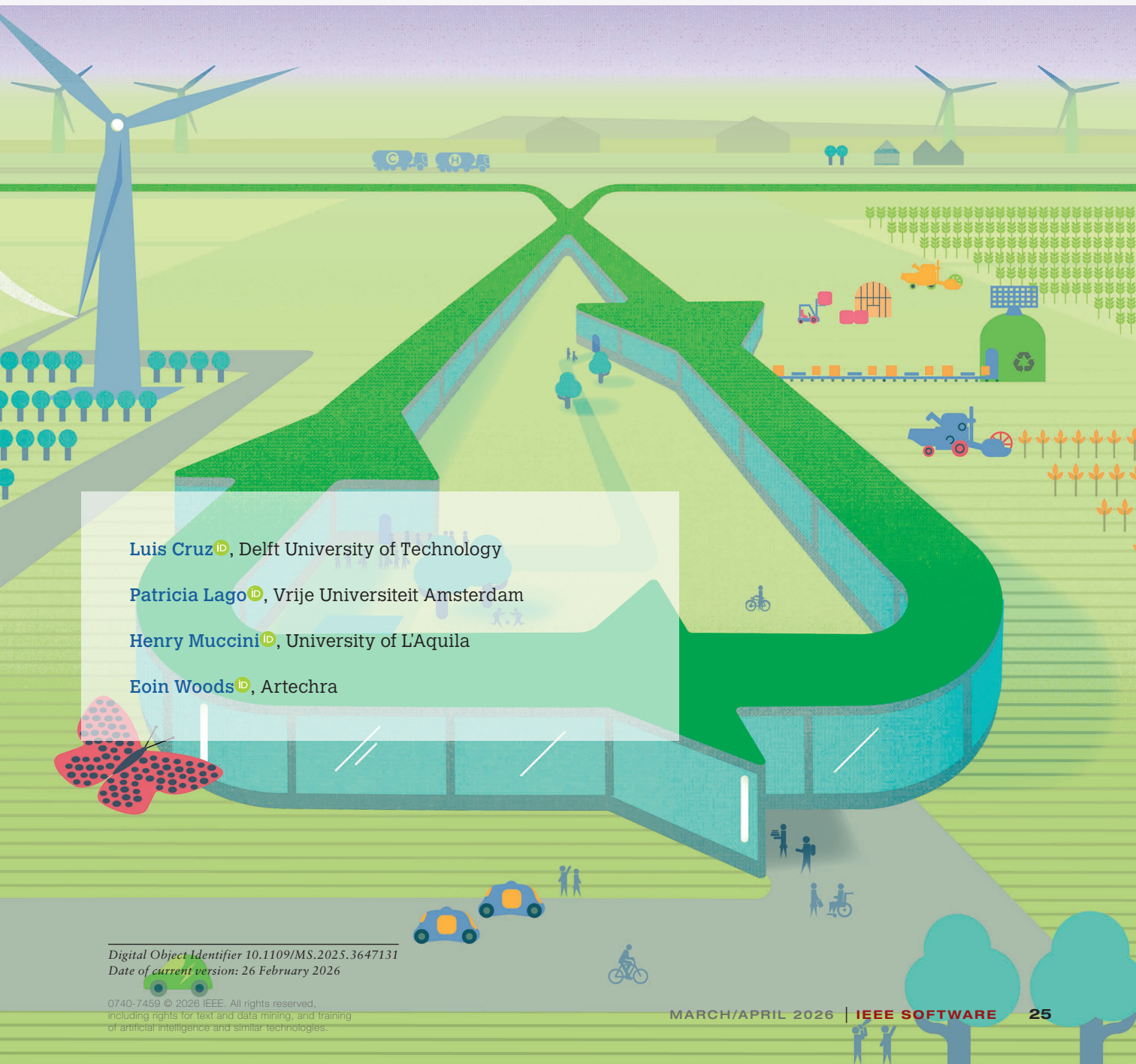
Green Clean Software Sustainability



Luis Cruz, Delft University of Technology

Patricia Lago, Vrije Universiteit Amsterdam

Henry Muccini, University of L'Aquila

Eoin Woods, Artechra

Digital Object Identifier 10.1109/MS.2025.3647131
Date of current version: 26 February 2026

THE ENERGY FOOTPRINT of software-intensive systems poses a significant concern. Energy-hungry software, such as blockchain applications and cryptocurrencies, the pervasive integration and usage of central cloud and edge services and applications, along with AI-enabled systems, contribute to this issue. To better understand the scale of the problem and the fact that this is not getting better, let us consider a few examples: GPT-4 training required approximately 5 gigawatt hour (GWh) (including computing-only estimates),[a] and up to 50 GWh when considering the full data center overhead and all GPUs used for the training[b] with an energy inference cost close to ~0.34 Wh per query. GPT-5, comparatively, required an estimate of 3.5 GWh for training (including computing-only estimates) and can consume 18–40 Wh per query (for a medium-length GPT-5 response of ~1,000 tokens), which is 50–100× higher than GPT-4.[c] As another example, Bitcoin's total annual energy consumption reached 173 TWh in 2025 with a crypto energy usage per transaction of approximately 1,135 kilowatt hour (kWh) compared to the 84 kWh per transaction used by Ethereum.[d] In addition, the global digital transformation of all industry sectors is accelerating the steep increase in software energy demands. All this has a direct impact on the energy consumption of data centers, estimated to be 415 TWh in 2024 and expected to grow globally to 945 TWh per year by 2030.[e]

> Reducing the carbon footprint must go hand in hand with minimizing the energy footprint.

Green clean software pertains to the minimization of the energy needed to execute and use software-intensive systems. Adopting renewable energy resources to "feed" software execution is simply not enough, so reducing the carbon footprint must go hand in hand with minimizing the energy footprint. On the other hand, software-intensive systems may be used to support green processes that aim to reduce the environmental impact of the sector, and, in fact, of any industry sector, of society, and planet Earth. Examples include software supporting the production and consumption of renewable energy resources, smart software for green-oriented behavioral change (e.g., adopting green public transportation and sustainable work practices), and the combination of energy optimization and digitalization (so-called twin transition). In addition, software sustainability from an environmental perspective may also concern software engineering and its processes: the energy used to develop, evolve, and maintain software-intensive systems is nonnegligible and also needs to be addressed.

A New Generation of Software Professionals

While green software and its engineering primarily target energy optimization, they fall under the general notion of Software Sustainability, which we define as "*the preservation of the long-term and beneficial use of software, and its appropriate evolution, in a context that continuously changes.*"[1] As such, software sustainability is a "must have" skill for any professional role making decisions that pertain to digitalization. Also, while it is not merely limited to *green clean*, software energy footprint is a global concern, hence a very good place to start with.

As others have identified,[2] sustainability is not just a question of environmental sustainability. While this is important, it is not sufficient to produce truly sustainable software systems. To achieve real long-term change in how we deliver software, we need to simultaneously consider the variation over time and the relative contextual change of software engineering related aspects in (at least) four distinct dimensions of sustainability, namely technical, economic, social, and environmental. We explain these dimensions of focus in the sidebar "Focus, Time, and Context in Software Sustainability."

[a] https://www.abcmoney.co.uk/2025/06/gpt-5-development-sparks-fresh-debate-over-ais-carbon-footprint ; At the moment of writing, OpenAI has not publicly disclosed the energy consumption associated with training GPT-4; consequently, reported figures are third-party estimates that vary with hardware/utilization/power usage utilization assumptions

[b] https://en.greentimes.ma/chatgpt-one-training-session-equals-three-days-of-a-citys-consumption

[c] https://en.greentimes.ma/chatgpt-one-training-session-equals-three-days-of-a-citys-consumption

[d] https://coinlaw.io/bitcoin-energy-consumption-statistics

[e] https://www.iea.org/reports/energy-and-ai/executive-summary

FOCUS, TIME, AND CONTEXT IN SOFTWARE SUSTAINABILITY

Based on the work by Lago and Malavolta,[2] we summarize here the basic definitions for the three software sustainability facets framing the dimensions of focus, variation over time, and contextual change.

DIMENSIONS OF FOCUS

The *technical* dimension of sustainability considers all of the activities involved in envisioning, building, deploying, operating, and ultimately decommissioning the hardware and software required by a system.

The *economic* dimension considers the financial aspects of delivering and operating software systems, including investment, cash flow, return on investment, operational expenditure, and capital expenditure and allocation.

When we talk about the *social* dimension, we look beyond our normal software engineering horizons to consider how our software systems integrate into and affect our communities and the resulting impact on individuals, families, social groups, and wider society.

Finally, the *environmental* dimension starts with energy consumption and greenhouse gas emissions but looks beyond this to understand, mitigate, and account for the broader ecological implications of a software system.

VARIATION OVER TIME

Based on the work by Hilty and Aebischer,[S1] we consider three types of sustainability-relevant effects on Information and Communication Technology (ICT) over time.

The *direct effects* pertain to the production, use or disposal of ICT itself. For software products or services, for instance, these may consider the sustainability of software engineering processes or the sustainability impact of their execution after deployment.

The *enabling effects* of ICT are the result of applying ICT services in different contexts and can be grouped into induction (when ICT stimulates consumption), obsolescence (when ICT shortens the useful life of another resource), substitution (when ICT replaces the use of another resource) and optimization (when using ICT reduces the use of another resource) effects. Again, in software engineering, we may consider the sustainability impact of the processes supported by the software.

Finally, the *systemic effects* are the long-term socio-economic effects that induce a behavioral or economic structural change. For example, if introducing software in some industry sector changes the needed digital competencies and skills of the workforce for that sector.

CONTEXTUAL CHANGE

According to Khabou et al.,[S2] context *"describes relevant aspects of the surrounding physical and computing environments, such as the location and the activity of the user."*

Software practitioners and researchers regularly mention context as essential in software engineering and decision making, but they rarely make it explicit. This is why the notion of context and its elements, is less understood. As such, for the time being, we hypothesize that context elements can be classified in the four dimensions of focus. For example, we may have context elements that are technical (e.g., network devices), economic (e.g., budget constraints), social (e.g., stakeholders), or environmental (e.g., energy resources).

References

S1. L. M. Hilty and B. Aebischer, "ICT for sustainability: An emerging research field," in *Proc. ICT Innov. Sustainability*, 2014, pp. 3–36, doi: 10.1007/978-3-319-09228-7_1.

S2. N. Khabou, I. B. Rodriguez, and M. Jmaiel, "An overview of a novel analysis approach for enhancing context awareness in smart environments," *Inf. Softw. Technol.*, vol. 111, Jul. 2019, pp. 131–143, doi: 10.1016/j.infsof.2018.11.001.

It is by working out how to consider and balance all of these dimensions of sustainability in systems development and operation that we will create truly sustainable software systems and create a new generation of practitioners with the skills to achieve this.

We illustrate how our vision of a tri-fold sustainability (i.e., in terms of dimensions of focus, time and context) works by using the articles in this special issue as examples (see Table 1). In Table 1, we summarize the type of coverage for each of the three dimensions (focus, time, and context) if present. Also, we use a dashed line if the linked dimension

TABLE 1. Sustainability coverage by the articles in this special issue (dashed lines indicate weak coverage).

Article #	Description	Focus				Time			Context			
		TEC	ECO	ENV	SOC	DIR	IND	SYS	TEC	ECO	ENV	SOC
A1	Optimization of the energy consumption (ENV) of the AI pipeline (TEC) at runtime (DIR). Context is the execution environment (TEC); key takeaways and findings for stakeholder responsibility are summarized (SOC).	●		●	●	●			●		●	●
A2	Evaluation of the energy efficiency (ENV) of code which is ChatGPT-generated (TEC) versus human-written (SOC). Context is the software development process with the human in the loop (SOC - TEC).	●		●	●	●			●		●	●
A3	Cloud predeployment estimation of energy and carbon footprint (ENV) of workloads (TEC) at design and runtime (DIR), which enables carbon-aware architecture design decisions. Context is the cloud (TEC) and estimation of carbon intensity (ENV). Implications for research and practice are summarized (SOC).	●		●	●	●			●		●	●
A4	Comparison of the energy consumption (ENV) of legacy versus modernized web applications (TEC) in industrial contexts (TEC), with some recommendations for IT professionals (SOC).	●		●	●	●			●		●	●
A5	A method to provide carbon estimates (ENV) of network devices (TEC) near real-time (DIR). Context is being considered as the carbon intensity (ENV) where devices (TEC) are geolocated.	●		●		●			●		●	
A6	Definition of a context-independent metric quantifying performance (TEC) versus energy consumption (ENV) in the video gaming domain/context (TEC). The metric is foreseen to be reusable as a de facto standard (SOC).	●		●	●	●			●		●	●

TEC: technical; ECO: economic; ENV: environmental; SOC: social; DIR: direct; IND: indirect; SYS: systemic.

is only weakly addressed. For example, "LEAF: A Layered Emission Assessment Framework for Cloud Deployments"[A3] provides design-time estimates of carbon intensity by using regional generalizations, which do not necessarily reflect the actual in-context runtime energy mix. It also reports some implications for research and practice, which are a good start but to be truly adopted would need actionable reframing, e.g., in the form of ADRs[f] and/or a professional open catalog.

Based on the summary in Table 1 and our reflection, we draw some considerations for a new generation of software professionals that wish to integrate sustainability in their daily work.

[f]Architectural Decision Records, https://adr.github.io

We should invest more actively in leveraging the precious takeaways as reusable know-hows for IT professionals. Software exists for a (social) purpose; that is, it should be useful for something or to someone. The social coverage in the "Focus" column of Table 1 shows that such *beneficial use of software* (quoting our definition of software sustainability) is often left implicit and rarely covered

(here only by "Is ChatGPT Generated Code Really Green? Evaluating AI-Generated Solutions for Energy-Efficient Coding Practices,"[A2] which considers *the benefits of the machine for the human developer*). On a positive note, research results often do contribute knowledge or know-how that could (and should) be leveraged as actionable guidelines or practices. This is reflected also in most articles in this special issue (see again the social coverage in column "Context"). Here, the weak coverage represented by the dashed lines is a call for action to extract and disseminate such actionable guidelines and practices.

In the time dimension, only the direct impact (e.g., at runtime) of green clean software is being considered. Table 1 clearly illustrates how all articles of this special issue cover only the direct impact in the "Time" dimension. While useful, this is unfortunately just a prerequisite for true sustainability. For example, there is widespread concern in the industry about embodied emissions, which is an indirect environmental effect of information and communication technology (ICT). In other words, we must indeed measure the energy consumption of running software-intensive systems, but in addition we should also adopt practices (i.e., indirect impact) that lead over time to lesser/optimized/responsible consumption, and ultimately to a cultural change (i.e., systemic impact) of sufficient[3] and optimized use of software-intensive systems.

Context matters and yet it is largely neglected. Overall, we observe over and over again, that context is regularly mentioned in general terms, but the contextual elements that influence software sustainability are not made explicit nor used to qualify the contribution proposed in the articles. (This is, by the way, often the case in practice, too.) If we would link a sustainability result or measure to the related contextual elements, a change in the context could be traced back to the possible positive or negative implications for the sustainability result.

Overview of the Special Issue Articles

This special issue covers both the *green clean software,* and the *green through software* dimensions, with special emphasis on the role played by green software engineering. After a rigorous review process, six articles were accepted for this special issue. Next, we provide a brief summary of each.

Tu(r)ning AI Green: Exploring Energy Efficiency Cascading with Orthogonal Optimizations[A1]

AI is transforming how we engineer software systems, but its rapid growth comes with a substantial environmental footprint. This article introduces a systematic, software engineering approach to energy-efficient AI, showing that combining optimization techniques across five stages of the AI development pipeline—data, model architecture, training, system design, and inference—can produce cumulative energy savings of 4.6% to 94.6% with minimal impact on accuracy. The results provide practical guidance for practitioners to balance efficiency, performance, and environmental responsibility in the development of AI systems.

Is ChatGPT Really Green? Evaluating AI-Generated Code for Energy-Efficiency[A2]

Building on the theme of AI sustainability, this study evaluates the energy efficiency of AI-generated code. By comparing ChatGPT responses to 200 Stack Overflow questions with accepted human answers, the authors identify 40 cases where human solutions outperform AI in energy efficiency. This work highlights where human insight remains critical, while offering practical guidance for designing energy-aware AI coding assistants, demonstrating how AI and human collaboration can drive greener software practices.

LEAF: A Layered Emission Assessment Framework for Cloud Deployments[A3]

Extending the discussion from software design to infrastructure, the LEAF framework enables early-stage carbon footprint estimation for cloud deployments. By analyzing infrastructure-as-code configurations and simulating workloads, LEAF provides predeployment energy estimates that align within 15.7% of post-deployment measurements. This approach empowers architects and developers to make sustainability-informed decisions before provisioning resources, bridging the gap between software practices and infrastructure planning.

Energy Consumption of Web Applications: Measurement Challenges in Practice[A4]

The focus then shifts to industrial applications, where measuring and interpreting energy use in complex web systems presents unique challenges. Using monitoring tools and workload simulations at Berger-Levrault, the study demonstrates how architectural choices and technology stacks can reduce energy consumption by up to 70%. These findings reinforce the need for usage-aware design decisions and provide practical recommendations for applying green software principles in real-world, large-scale industrial environments.

NetCarbTrace: A Probing Tool to Measure and Explore the Carbon Footprint of Computer Networks[A5]

Beyond applications and infrastructure, energy efficiency must also consider network systems. The NetCarbTrace tool offers a context-aware method to assess the energy and carbon footprint of network devices, accounting for their nonlinear power profiles. By analyzing real-world endpoints, this study demonstrates near real-time carbon estimation and provides actionable insights for designing carbon-optimized network services, emphasizing that sustainability must consider the entire software ecosystem.

Energy Profiling in Games: Introducing a Frame-Based Power Consumption Metric[A6]

Finally, the issue explores energy efficiency in digital entertainment. By introducing a generic metric in joules per frame, the study provides a standardized way to quantify the energy consumption of video games, allowing developers to balance performance and environmental impact. This work extends sustainable software practices into consumer-facing applications, highlighting the broad applicability of green software engineering principles across diverse domains.

We hope this special theme issue will inspire our readers to adopt the findings from the included articles, and the considerations we draw from them as well as our experience and personal reflection.

Acknowledgment

We would like to extend our sincere gratitude to all reviewers who donated their time and provided constructive feedback to the authors and to Joran Leest for contributing to the sustainability visualization.

References

1. P. Lago, "The digital society is already here – Pity it is 'unsustainable'," in *Connected World: Insights from 100 Academics on How to Build Better Connections*, I. Vermeulen, Ed., Amsterdam, The Netherlands: VU Univ. Press, 2023, pp. 55–58. [Online]. Available: https://vuuniversitypress.com/product/connected-world
2. P. Lago and I. Malavolta, "A new vision on software sustainability and its engineering," *IEEE Softw.*, vol. 43, no. 1, pp. 119–123, Jan./Feb. 2026, doi: 10.1109/MS.2025.3622804.
3. M. Madon and P. Lago, "'We are always on, is that really necessary?' Exploring the path to digital sufficiency in flexible work," in *Proc. Int. Conf. ICT Sustainability (ICT4S)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 24–34, doi: 10.1109/ICT4S58814.2023.00012.

Appendix: Related Articles

A1. S. Rajput, M. Saad, and T. Sharma, "Tu(r)ning AI green: Exploring energy efficiency cascading with orthogonal optimizations," *IEEE Softw.*, vol. 43, no. 2, pp. 32–41, Mar./Apr.

ABOUT THE AUTHORS

LUÍS CRUZ is an assistant professor in software engineering at Delft University of Technology and co-founder of GreenSeal.dev, 2600 AA Delft, The Netherlands. Contact him at l.cruz@tudelft.nl.

PATRICIA LAGO is a professor of software engineering in the Software and Sustainability research group and director of the Digital Sustainability Center at Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands. Contact her at p.lago@vu.nl.

HENRY MUCCINI is a professor of software engineering in the SWEN research group and leader of the FrAmeLab research laboratory at the University of L'Aquila, 67100 L'Aquila, Italy. Contact him at henry.muccini@univaq.it.

EOIN WOODS is an independent consultant in the fields of software architecture, green software, and software engineering at Artechra, TN1 2QP Tunbridge Wells, U.K. Contact him at eoin.woods@artechra.com.

2026, doi: 10.1109/MS.2025.3645090.

A2. A. Swaraj and S. Kumar, "Is ChatGPT generated code really green? Evaluating AI-generated solutions for energy-efficient coding practices," *IEEE Softw.*, vol. 43, no. 2, pp. 42–50, Mar./Apr. 2026, doi: 10.1109/MS.2025.3644903.

A3. P. Singhal et al., "LEAF: A layered emission assessment framework for cloud deployments," *IEEE Softw.*, vol. 43, no. 2, pp. 51–60, Mar./Apr. 2026, doi: 10.1109/MS.2025.3637338.

A4. L. Khrouf, A. Shatnawi, R. Rouvoy, B. Verhaeghe, and B. T. Niang, "Energy consumption of web applications: Measurement challenges in practice," *IEEE Softw.*, vol. 43, no. 2, pp. 61–69, Mar./Apr. 2026, doi: 10.1109/MS.2025.3641504.

A5. R. Vergallo, A. Campa, S. Casciaro, G. Ciccarese, A. Mongelli, and L. Mainetti, "NetCarbTrace: A probing tool to measure and explore the carbon footprint of computer networks," *IEEE Softw.*, vol. 43, no. 2, pp. 70–77, Mar./Apr. 2026, doi: 10.1109/MS.2025.3636577.

A6. L. Lou, L. Guichard, V. Plantevin, H. B. Abdessalem, and Y. Francillette, "Energy profiling in games: Introducing a frame-based power consumption metric," *IEEE Softw.*, vol. 43, no. 2, pp. 78–84, Mar./Apr. 2026, doi: 10.1109/MS.2025.3646195.

Digital Object Identifier 10.1109/MS.2026.3659403