Real-Time Systems (2025) 61:294–299
https://doi.org/10.1007/s11241-025-09440-0

To MILP or not to MILP? On AI techniques for the design and optimization of real-time systems

Daniel Casini[1]

Accepted: 9 May 2025 / Published online: 26 May 2025

Abstract

Artificial intelligence (AI) is becoming increasingly relevant in many contexts. In embedded real-time systems, most of the previous research has focused on real-time guarantees for AI workloads (RT-for-AI). Instead, this position paper discusses the potential benefits and application cases of the complementary direction of using AI to optimize real-time systems themselves (AI-for-RT). It presents a vision where AI techniques, such as supervised and reinforcement learning, support system design and online configuration activities that are traditionally addressed using Mixed-Integer Linear Programming (MILP) or heuristic methods. The paper discusses scenarios where AI can potentially outperform classical techniques—such as recursive real-time analysis, systems with complex hardware/software interactions, and dynamic resource management—highlighting the promise of AI in both design-time and runtime real-time systems optimization. Solutions are left to future work: the goal is to populate the "Roadmap Towards Learning-Enabled and Learning-Assisted Real-Time Systems", which is the target of this special issue.

Keywords Real-time systems · Machine learning · System design

1 Introduction

Artificial intelligence (AI) is revolutionizing the world of computer systems. Embedded real-time systems are not an exception: the incredible performance given by AI-based techniques are making them both a target use case and a means for improving real-time systems. When AI applications are part of the system that needs to respond with predictable timing constraints, they correspond to a special workload to be modeled, analyzed, optimized, and supported by the real-time community to satisfy timing bounds. This side of the problem, which can be called RT-for-AI, has largely been addressed by the community since the early stages of the AI revolution.

✉ Daniel Casini
daniel.casini@santannapisa.it

[1] Sant'Anna School of Advanced Studies, Pisa, Italy

This paper. This paper highlights the other side of the problem: AI-for-RT, meaning the use of AI (and machine learning in general) to address the challenges arising in the real-time systems domain. This paper particularly emphasizes those due to the design and optimization of real-time systems that are difficult or even impossible to suitably optimize with standard techniques, such as Mixed-Integer Linear Programming (MILP) formulations, and online decision-making problems. Examples are provided.

Related work. Some work targeting AI-for-RT (and ML-for-RT) has already been done by the community.[1] Vădineanu and Nasri (2020) explored the use of regression-based machine learning for the period inference in the presence of uncertainty in the parameters. Lee et al. developed a ML framework to assign priorities under global fixed-priority scheduling by using a pointer network as a base neural architecture (supervised learning) (Lee et al. 2021) and reinforcement learning (unsupervised learning) (Lee et al. 2020). Other work (Amalou et al. 2023) targeted the estimation of worst-case execution times using transformers.

2 AI-driven system design

The Problem. The problem we propose to address is the system design and configuration of a system subject to timing constraints leveraging AI techniques. The problems to be solved range from the task-to-core assignment, priorities, and parameter configurations in general to satisfy timing constraints or meeting high-level Key Performance Indicators (KPI), which are more extensively discussed in the following.

To MILP, or not to MILP? The reader who is familiar with real-time systems research may easily recognize that, in many cases, this problem has been very successfully solved by leveraging formulations as an optimization problem, e.g., using MILP. While it is true that whenever the problem is suitable to be formulated in a linear manner, MILP is a great optimization solution, since it guarantees to output the optimal solution to the encoded problem if enough time is provided, it has several limitations. Some MILPs may have scalability issues depending on the input space: in some cases, LP (linear programming) approaches may be devised to run faster whenever it is possible to encode the problem without using binary variables. Nevertheless, in many cases, even when it is possible to encode the problem as a MILP, this comes at the expense of "linearizing" some parts, most commonly the schedulability test, often with the price of turning a necessary and sufficient test into a sufficient-only test. This is often the case for real-time analysis techniques whenever the optimization procedure also needs to account for the scalability and time required to solve the problem. Hence, in these cases, the MILP returns an optimal solution to an approximated problem, which is, anyway, an optimized solution to the original problem.

The vision of AI-based real-time systems design. The challenges above are often solved by applying far simpler heuristic algorithms (e.g., think to best-fit or worst-fit

---

[1] The discussion of related papers is necessarily not comprehensive due to the tight space constraints.

heuristics for the bin-packing problem) that, on the one hand, do not have to be linearized, and they can directly adopt the original, potentially non-linear, response-time analysis technique but without providing any guarantee of finding the optimal solution or any optimality gap (distance from the optimal solution, typically reported by MILP solvers). Hence, using AI techniques such as reinforcement learning and supervised learning based on neural networks, which have proven to perform incredibly well in many contexts, could be a great fit to optimize complex real-time systems. The significant performance improvements achieved in other contexts suggest that this is also a flourishing research direction that this is also a promising research direction to improve the design of real-time systems. Furthermore, other machine learning techniques, such as random forest, could apply very well to solve these design problems.

A possible way of looking at the problem is shown in Fig. 1. An “AI-based design block”, i.e., an AI-based optimizer, is linked in series to a “Traditional Verifier Block”, providing one or more solutions. As is common in artificial intelligence, the AI-based block could provide the best-performing solutions, e.g., the top 5. The “Traditional Verifier Block” implements traditional real-time analysis techniques to determine the feasibility of the provided solutions according to the timing (and possibly other non-functional) requirements. A feedback must be provided to the AI block in case none of the provided solutions guarantees the non-functional constraints.

3 Design problems

This section summarizes some relevant cases in which the system to be optimized is complex, or the corresponding real-time analysis is highly unsuitable to be encoded with traditional methods, and that could benefit from AI-based optimization.

When the real-time analysis is a recursive algorithm. Prior work targeted the analysis of parallel tasks modeled as DAGs under partitioned scheduling, meaning that each vertex of each DAG task can be statically assigned to a different core and never migrate. Response-time analysis techniques under this setting (Fonseca et al. 2016; Casini et al. 2018) have been devised by leveraging analysis techniques for

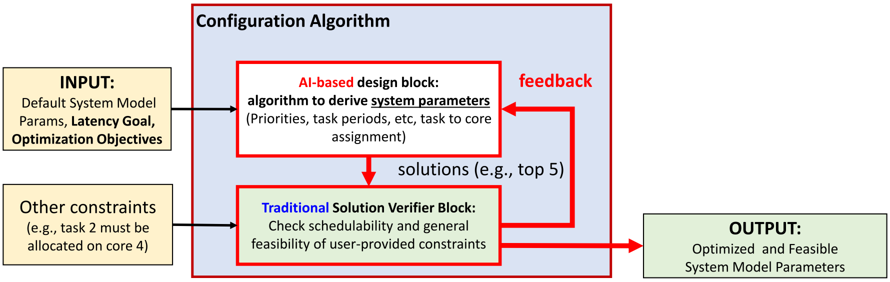
Configuration Algorithm
INPUT:
Default System Model Params, Latency Goal, Optimization Objectives
Other constraints (e.g., task 2 must be allocated on core 4)
AI-based design block: algorithm to derive system parameters (Priorities, task periods, etc, task to core assignment)
feedback
solutions (e.g., top 5)
Traditional Solution Verifier Block: Check schedulability and general feasibility of user-provided constraints
OUTPUT:
Optimized and Feasible System Model Parameters

Fig. 1 AI for the design of real-time systems

self-suspending tasks (Chen et al. 2019). The mapping is made as follows: by considering the perspective of each specific core, vertexes of the path that executes on the current core can be seen as execution regions of a segmented self-suspending task; vertexes in which execution is on a different core are then suspension regions. Each path is studied separately, applying a *recursive algorithm* to divide a path into smaller subpaths to be considered for calculating the suspension time. The base case of the recursion is when the analyzed subpath has a single node, i.e., it does not have suspensions, and a response time can be computed and propagated back as the suspension time of other subpaths. Clearly, a recursive algorithm as a real-time analysis technique is a case in which a corresponding design configuration problem can hardly be encoded as a traditional optimization problem, and AI techniques can be of great help.

When the real-time analysis is a MILP. There are cases in which the real-time analysis itself is formulated as an optimization problem (e.g., a MILP). Some relevant examples are the case of bundled gang real-time tasks (Wasly and Pellizzoni 2019; Rispo et al. 2024), weakly-hard analysis techniques under the (m,k) model (Pazzaglia et al. 2020), and memory-CPU coscheduling problems (Casini et al. 2020). In these cases, the optimization problems often aim at maximizing the interference while creating a tentative schedule. The constraints encode the scheduling rules, i.e., how interference can be spread among different computational activities.

When the analysis involves complex software stacks and fine-grained hardware effects. Another relevant case is when the analysis comprises multiple scheduling effects, middleware, e.g., ROS 2, the DDS, and scheduling delays due to the operating systems. This is likely to be relevant in any system running Autoware (Kato et al. 2018), a popular ROS-based autonomous driving framework. For these cases, the analysis method often relies on the resolution of complex inter-task dependencies via a global search of a fixed point. Further complexity arises when the analysis of processing workloads must be integrated with fine-grained memory-contention analysis, which, in turn, can be formulated as MILP optimization problems (Casini et al. 2020; Hassan and Pellizzoni 2020).

Online decision-making to optimize real-time constraints. Another, different, configuration problem, is the online configuration of a complex distributed system composed of multiple hierarchical orchestrators that allocate resources at runtime. For example, at a higher level, a coarse-grained orchestrator assigns physical resources (e.g., nodes) to other, low-level orchestrators, which have a fine-grained control of the allocation of resources. A similar architecture is envisioned in the NANCY European Project (2025). In this case, a time-sensitive domain (and the corresponding orchestrator) leverages the SCHED_DEADLINE scheduler of Linux to provide fine-grained virtualization of the processing capacity and still providing timing isolation. A description of the reference system is reported in Abeni et al. (2014). The time-sensitive orchestrator is in charge of mapping high-level KPI (e.g., “overall network service latency” <= ...) in a scheduling decision that assigns to a certain container, encapsulated in a SCHED_DEADLINE reservation, to a certain number of cores, with a budget and a period parameter. However, the high-level KPI is affected by multiple factors (e.g., network latency, virtualization overheads). How to leverage these inputs to assign proper SCHED_DEADLINE parameters is an exciting research direction.

4 Conclusions

This paper discussed some target problems in which AI-based optimization is likely to be useful and effective for the design of real-time systems. It adds two problems to the list of open problems in the "Roadmap Towards Learning-Enabled and Learning-Assisted Real-Time Systems" which is the target of this special issue: (1) the use of AI for the design-time optimization of complex real-time systems to guarantee real-time constraints and (2) the use of AI in online runtime configuration algorithms that need to guarantee high-level KPIs. Future work will target the solutions to these problems. For example, an interesting first step would be to compare whether supervised or unsupervised (e.g., reinforcement learning) is most effective for the target purposes.

Author Contributions D.C., as the only author, did everything.

Funding Open access funding provided by Scuola Superiore Sant'Anna within the CRUI-CARE Agreement.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

Abeni L, Cucinotta T, Casini D (2024) Period estimation for linux-based edge computing virtualization with strong temporal isolation. In: 2024 IEEE 3rd Real-Time and Intelligent Edge Computing Workshop (RAGE), pp. 1–6. IEEE

Amalou AN, Fromont E, Puaut I (2023) Cawet: Context-aware worst-case execution time estimation using transformers. In: ECRTS 2023-35th Euromicro Conference on Real-Time Systems, vol. 262, pp. 7–1. Schloss Dagstuhl-Leibniz-Zentrum für Informatik

Casini D, Biondi A, Nelissen G, Buttazzo G (2018) Partitioned fixed-priority scheduling of parallel tasks without preemptions. In: 2018 IEEE Real-Time Systems Symposium (RTSS), pp. 421–433. IEEE

Casini D, Biondi A, Nelissen G, Buttazzo G (2020) A holistic memory contention analysis for parallel real-time tasks under partitioned scheduling. In: 2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 239–252. IEEE

Casini D, Pazzaglia P, Biondi A, Di Natale M, Buttazzo G (2020) Predictable memory-cpu co-scheduling with support for latency-sensitive tasks. In: Proceedings of the 57th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, pp. 1–6

Chen J-J, Nelissen G, Huang W-H, Yang M, Brandenburg B, Bletsas K, Liu C, Richard P, Ridouard F, Audsley N et al (2019) Many suspensions, many problems: a review of self-suspending tasks in real-time systems. Real-Time Syst 55(1):144–207

Fonseca J, Nelissen G, Nelis V, Pinho LM (2016) Response time analysis of sporadic dag tasks under partitioned scheduling. In: 2016 11th IEEE Symposium on Industrial Embedded Systems (SIES), pp. 1–10

Hassan M, Pellizzoni R (2020) Analysis of memory-contention in heterogeneous COTS mpsocs. In: Völp M (ed.) 32nd Euromicro Conference on Real-Time Systems, ECRTS 2020, July 7–10, Virtual Conference. LIPIcs, vol. 165, pp. 23–12324. Schloss Dagstuhl - Leibniz-Zentrum für Informatik

Kato S, Tokunaga S, Maruyama Y, Maeda S, Hirabayashi M, Kitsukawa Y, Monrroy A, Ando T, Fujii Y, Azumi T (2018) Autoware on board: Enabling autonomous vehicles with embedded systems. In: 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), pp. 287–296

Lee H, Lee J, Yeom I, Woo H (2020) Panda: reinforcement learning-based priority assignment for multi-processor real-time scheduling. IEEE Access 8:185570–185583

Lee S, Baek H, Woo H, Shin KG, Lee J (2021) ML for RT: Priority assignment using machine learning. In: Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 118–130

NANCY European Project (2025) A secure and intelligent architecture for the beyond the fifth generation (B5G) wireless network - European Project. https://nancy-project.eu/. Accessed: 2025-03-18

Pazzaglia P, Sun Y, Natale MD (2020) Generalized weakly hard schedulability analysis for real-time periodic tasks. ACM Trans Embed Comput Syst 20(1):1–26

Rispo V, Aromolo F, Casini D, Biondi A (2024) Response-time analysis of bundled gang tasks under partitioned fp scheduling. IEEE Trans Comput 73(11):2534–2547

Vădineanu S, Nasri M (2020) Robust and accurate period inference using regression-based techniques. In: Proceedings of the IEEE Real-Time Systems Symposium (RTSS), pp. 364–376

Wasly S, Pellizzoni R (2019) Bundled scheduling of parallel real-time tasks. In: 2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 130–142

Daniel Casini is a Tenure-Track Assistant Professor at the Real-Time Systems (ReTiS) Laboratory of the Scuola Superiore Sant'Anna of Pisa. He graduated (cum laude) in Embedded Computing Systems Engineering, a Master degree jointly offered by the Scuola Superiore Sant'Anna of Pisa and University of Pisa, and received a Ph.D. in computer engineering at the Scuola Superiore Sant'Anna of Pisa (with honors), working under the supervision of Prof. Alessandro Biondi and Prof. Giorgio Buttazzo. In 2019, he has been visiting scholar at the Max Planck Institute for Software Systems (Germany). His research interests include software predictability in multi-processor systems, schedulability analysis, synchronization protocols, and the design and implementation of real-time operating systems and hypervisors.