Learning-based Response Time Analysis in Real-Time Embedded Systems: A Simulation-based Approach

Mahshid Helali Moghadam[1,3], Mehrdad Saadatmand[1], Markus Borg[2], Markus Bohlin[1], Björn Lisper[3]
[1]RISE SICS Västerås, SE-722 13, Västerås
[2]RISE SICS Lund, SE-223 70, Lund
[3]Mälardalen University, SE-721 23, Västerås
Sweden
{mahshid.helali.moghadam, mehrdad.saadatmand, markus.borg, markus.bohlin}@ri.se, bjorn.lisper@mdh.se

ABSTRACT

Response time analysis is an essential task to verify the behavior of real-time systems. Several response time analysis methods have been proposed to address this challenge, particularly for real-time systems with different levels of complexity. Static analysis is a popular approach in this context, but its practical applicability is limited due to the high complexity of the industrial real-time systems, as well as many unpredictable runtime events in these systems. In this work-in-progress paper, we propose a simulation-based response time analysis approach using reinforcement learning to find the execution scenarios leading to the worst-case response time. The approach learns how to provide a practical estimation of the worst-case response time through simulating the program without performing static analysis. Our initial study suggests that the proposed approach could be applicable in the simulation environments of the industrial real-time control systems to provide a practical estimation of the execution scenarios leading to the worst-case response time.[1]

CCS CONCEPTS

• **Software and its engineering** → **Software performance;** • **Computer systems organization** → Real-time systems; • **Computing methodologies** → Machine learning;

KEYWORDS

Response time analysis; Worst-case response time; Real-time embedded systems; Reinforcement learning; Simulation

ACM Reference format:

M. H. Moghadam, M. Saadatmand, M. Borg, M. Bohlin, B. Lisper. 2018. Learning-based Response Time Analysis in Real-Time Embedded Systems: A Simulation-based Approach. In *Proceedings of IEEE/ACM 1st International Workshop on Software Qualities and their Dependencies, Gothenburg, Sweden, May 28, 2018 (SQUADE'18)*, 4 pages.

DOI: 10.1145/3194095.3194097

1 INTRODUCTION

Nowadays, embedded systems have become important parts of our life and are becoming more complex and powerful. Real-time programs running on embedded systems are key parts of many industrial control systems like those in telecommunication, railway, avionics, automotive, and medical care. Many of the industrial real-time embedded systems are referred to as complex real-time systems due to the internal functional complexity resulting from dependencies between tasks, asynchronous message-passing, runtime changeable priorities, and task offsets [1].

In general, there are strict timing requirements in terms of demand response times and deadlines, particularly for time-critical programs in safety-critical systems. Behavior correctness of these systems highly depends on both functional correctness and satisfaction of non-functional requirements like limits on acceptable response time. Worst-Case Response Time (WCRT) estimation is an essential task for verifying the behavior of the real-time systems.

Static analysis, measurement-based analysis, hybrid measurement-based analysis, parametric analysis, statistical analysis, simulation-based analysis and formal timing analysis are popular approaches to provide an estimation of the WCRT of a system [2-11]. Due to the functional complexity, the applicability of the theoretical static approaches which work based on the assumption of the task independence in the analysis, will be partially limited in the complex real-time systems.

In this paper, we propose a simulation-based WCRT analysis using Reinforcement Learning (RL). The proposed approach is a learning-based method to learn a policy finding the execution scenarios leading to the WCRT. We used Q-learning as a model-free reinforcement learning [12] to find the policy of estimating the

*Produces the permission block, and copyright information
†The full version of the author's guide is available as acmart.pdf document
[1]It is a datatype.

SQUADE'18:, May 28, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5737-1/18/05…$15.00
https://doi.org/10.1145/3194095.3194097

worst-case execution scenarios of the program and provide a practical estimation of the WCRT. Q-Learning does not need any precise models of the system and can be applied to all problems which can be modeled as a Markov Decision Process.

The rest of this paper is organized as follows; Section 2 discusses the motivation of using RL to estimate the WCRT. The technical details of the proposed approach, and a short description of its applicability are presented in Section 3. Section 4 provides a review of the related works and background approaches. Conclusion and future directions of this work-in-progress research are provided in Section 5.

2 MOTIVATION AND BACKGROUND

WCRT analysis as an essential part of the timing analysis could remain as a challenge, in particular for complex real-time embedded systems. Due to the functional complexity, conventional static analysis techniques might not be fully applicable to complex real-time systems. Measurement-based or simulation-based analysis are other alternatives to static WCRT analysis. Measurement-based approaches are time-consuming and partially infeasible for large systems. While simulation-based approaches reduce the analysis time by analyzing a simpler model of the system but still considering many detailed information on the behavior of the system. However, providing a model-driven estimation approach often requires precise knowledge of the system and also the execution environment. The behavior complexities of the systems are barriers which could motivate towards using model-free learning approaches.

Reinforcement learning [12] is a type of learning based on experimenting on the environment. In RL, the learning agent observes the state of the environment regularly. It takes an action randomly or greedily based on its experience, receives a reward signal of the environment and updates its achieved experience. The main purpose of the agent during the learning is to find a policy maximizing the expected long-term reward. Q-Learning [12] is a model-free RL algorithm during which the agent learns a utility function associated to pairs of states and actions. In this paper, we use Q-learning combined with the simulation-based technique to provide a learning-based approach to find the execution scenarios leading to the WCRT and provide a practical estimation thereof.

3 LEARNING-BASED RESPONSE TIME ANALYSIS

Analyzing the WCRT, particularly for time-critical applications realizing the safety use cases in industrial embedded systems is of great importance. In this section, we present the details of the proposed learning-based response time analysis. Real-time applications, particularly in distributed real-time embedded systems like Train Control Management System (TCMS) in the transportation domain consist of multiple components distributed on the underlying platform. Component-based representation such as Function Block Diagram (FBD) format for PLC-based programs, is a common way to demonstrate the function flow of the applications. Various runtime conditions could affect the response time and cause different deviations from the demand response time. A practical estimation of WCRT could provide useful information for the execution control procedure. Upon this information the control procedure could provide adaptive control operations to avoid total failures in the system.

The main objective of the proposed learning-based response time analysis is to provide a practical estimation of the WCRT of the program/application. It uses Q-learning in a simulation environment to learn a policy finding the execution scenarios leading to the WCRT in a given state. We present the details of the main steps of the learning-based approach as follows:

1) *State detection*. The estimator agent observes the state of the system continuously upon the components execution. Fig. 1 shows a sample component-based representation of a Brake-by-Wire system [13].

At each execution level, multiple components might be executed in parallel. Before starting each execution level, the agent detects the current state of the application. The state of the application, *s(t)*, is defined in terms of $\big(El(t), RT(t), \mu_{RT}(t)\big)$, a tuple consisting of the execution level, $El(t)$, and the fuzzy status of the response time until the current point, $(RT(t), \mu_{RT}(t))$. The status of the response time, *RT,* is classified into three classes, i.e. *Low, Normal and High* according to their compliance with target/demand response time, τ. We used fuzzy classification [14] as a soft labeling technique to avoid specifying sharp boundaries between classes. Fuzzification involves defining membership functions over the values of the response time. Each fuzzy set is defined as $I = \{(x, \ \mu_I(x) \mid 0 < x, \ x \in R\}$ where $\mu_I : x \to [0,1]$. Membership function specifies to which degree; each value belongs to the fuzzy set I. Fig. 2 shows the membership functions of fuzzy sets on RT values. Among the state tuples, the tuple with the maximum membership value specifies the state of the system.

2) *Suggested Events as Actions*. One of the main steps of Q-learning procedure is applying the action which the agent recommends based on its observation. In the proposed approach, the actions are the possible occurrences/events, based on the given state. The set of possible events could be defined based on the architecture and the redundancy structure of the system.

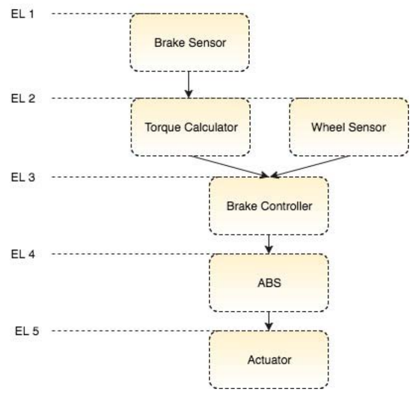

Figure 1: Component-based representation of a Brake-by-Wire system

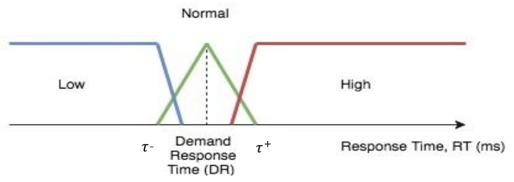

Figure 2: Fuzzy sets on response time values

Sample possible events for a distributed real-time control system could be defined as follows:

$$S_{TD} = \{TD_i|\ i = 1 \dots n\} \quad (1)$$

$$Events = \{S_{TD}, FD_1, \dots, FD_n, FC_1, \dots, FC_n\} \quad (2)$$

where S_{TD} represents the maximum delay in the output data channels. Each data communication channel has a time delay range. FD_i and FC_i represent occurring failure in output data channel i and component i respectively. Failures which do not lead to a total failure in the function are considered in the action set. The set of the events suggested by the agent are possible events at each state of the system. Many mission-critical real-time systems such as control systems in the transportation domain, use different types of redundancy techniques to provide availability and integrity against failures. Thus, in these systems, particularly for safety functions, single failure in the data transmission channels or the computation components could be tolerated.

3) *Computation of the reward signal.* According to the procedure of Q-learning algorithm, after applying the selected action, the system will go to the next state and after the transition, the agent receives a reward signal representing the effects of the applied action. In this approach, the recommended action is the suggested event that the agent selects to estimate the worst-case response time based on the current state of the system. Upon the selected event and based on its efficacy in estimating the worst-case response time, a reward signal is received by the agent which shows to which degree the suggested event in the current state could approach the worst-case response time. Regarding this objective, the reward signal has been defined as follows:

$$r(t) = \begin{cases} 0 & RT(t) \leq \tau^+ \\ \frac{RT(t)-DR}{\tau^+-DR} & \tau^+ \leq RT(t) \end{cases} \quad (3)$$

where DR is the demand response time and τ^+ is the upper boundary value of the Normal set around the demand response time.

In general, the main purpose of the Q-learning algorithm is to find a policy π, a mapping between the states of the system and the actions which leads to the maximum long-term reward. A utility value, $Q^\pi(s,a)$, which specifies the expected value of the achieved long-term reward, is associated with each pair of state and action. The expected long-term reward in each state and the associated utility value are defined as follows [12]:

$$R_t = r_{t+1} + \gamma r_{t+2} + \dots + \gamma^k r_{t+k+1} = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (4)$$

$$Q^\pi(s,a) = E^\pi[R_t|S_t = s, A_t = a] \quad (5)$$

Where $\gamma \in [0,1]$ is a discount factor which specifies to which degree the future rewards are weighted against the immediate reward.

The q-values, which are considered as the experience of the agent, keep the track of utility values for each pair of state and action. They are stored in a look-up table, *Q-Table* and used for deciding on action recommendations by the agent. Given the fuzzy nature of our state space, the q-values are updated according to the membership degree of the state using the following rule:

$$Q(s_t, a_t) = \mu_t^s[(1-\alpha)\, Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)] \quad (6)$$

Where $\alpha, 0 \leq \alpha \leq 1$ is the learning rate which controls the impacts of the new utility values on the q-values. The steps of the proposed approach using Q-learning are demonstrated in List 1. Through this approach, the agent learns which occurrences in each state could lead to the WCRT (maximum deviation from the demand response time). It finds a policy which selects the action maximizing the utility value for a given state, *s*, based on the following rule:

$$a(s) = \operatorname*{argmax}_{a'} Q(s, a') \quad (7)$$

Learning Performance. During the learning procedure, the agent could select actions randomly or based on the highest utility value. Different types of action selection strategies like *ε-greedy* and Softmax action selection could be employed to provide a trade-off between the exploration and exploitation of the action space. Tuning the learning parameters including learning rate α, and discount factor γ, could also affect the learning performance.

Applicability. Time-critical programs in many real-time control systems are often responsible for performing safety functions. Temporal behavior analysis, e.g., Worst-Case Response Time analysis is of great importance for these systems. For example, a violation from the demand response time in a Brake system, or an industrial robot in a production line or a time-critical function in an industrial control system could cause a fatal failure or huge loss in these systems. The proposed approach provides a simulation-based WCRT analysis using reinforcement learning. For each state, it suggests the event/occurrence which leads to the WCRT of the function. During the simulation, it finds a sequence of events resulting in the WCRT of the function. The proposed approach could be used in the simulation environment of the industrial control systems. It provides a practical estimation of the WCRT through simulation without performing static analysis. Thus, due to limitations of static analysis, it could perform better for complex real-time systems. However, the converged Q-table could be also used as input to further static analysis.

4 RELATED WORK

Satisfying the response time requirement as one of the main non-functional properties of real-time systems play a key role in the correctness of these systems. Response time analysis is essential for time-critical functions in real-time systems and also a major concern in many industries which employ real-time control systems. Several different techniques have been proposed for the WCRT analysis of real-time systems.

Algorithm: Learning-based Response Time Analysis

```
Required: S, A, ε, α, γ
Initialize q-values, $Q(s,a) = 0, \forall s \in S, \forall a \in A$
1. Observe the fuzzy state of the program, s(t)
2. Select an action/event using the action selection policy (e.g., ε-greedy)
3. Apply the selected action, let the system continue running and execute the next level of the components
4. Detect the new state of the system after executing the components regarding the selected event
5. Receive the reward signal, $r_{t+1}$
6. Update the q-value by
    $Q(s_t,a_t) = \mu_t^s[(1-\alpha)\,Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a)]$
7. Repeat for each newly observed state until the end of the program.
```

In general, the proposed approaches could be classified into multiple categories as follows:

Static analysis, measurement-based analysis, hybrid measurement-based analysis, parametric analysis, statistical analysis, simulation-based analysis and formal timing analysis are common approaches to provide an estimation of the WCRT of the system. In this context, the static analysis methods work based on analyzing the source code [2-3], while measurement-based and hybrid measurement-based methods perform analysis mostly based on running the program on the target hardware [4-5]. Parametric analysis works in a similar way to static analysis but uses parametric formula to express the WCRT [6]. Measurement-based analysis might be infeasible for large systems where running sufficient number of samples of the systems are time-consuming. While simulation-based analysis uses a model of the system with less complexity, but still containing the enough detailed information on the behavior of the system [8-10, 16]. Monte Carlo simulation [15] is a popular simulation-based method for WCRT analysis. Moreover, meta heuristics algorithms can be also used in combination with Monte Carlo simulation [8, 16]. Statistical analysis uses statistics theories to provide an estimation of WCRT based on a reference data set [7, 17]. Formal analysis like model checking has been used for WCRT analysis in case there is not a large system with high complexity [11].

This work-in-progress paper proposes a simulation-based method based on Q-learning to provide a practical estimation of the WCRT without having prior detailed knowledge of the system.

5 CONCLUSION

Estimating execution scenarios resulting in the WCRT and providing a practical estimation of the WCRT is a challenge for many industrial complex real-time systems. In this paper, we present a simulation-based analysis method to find the execution scenarios resulting in the WCRT using reinforcement learning. We applied Q-learning as a model-free learning algorithm to the simulation-based approach. It provides the optimal policy to find the sequences of the execution conditions resulting in the WCRT without having detailed information on the model of the system. Our next steps will be as follows: First, efficacy evaluation of the proposed approach on the simulation tools of the industrial real-time embedded systems. Second considering more complexity factors of the programs to provide more accurate models for the states, actions and the reward signal.

ACKNOWLEDGMENTS

This research has been funded by the ITEA3 initiative TESTOMAT Project (www.testomatproject.eu) through Vinnova and by KKS through the TOCSYC project.

REFERENCES

[1] Y. Lu, Approximation Techniques for Timing Analysis of Complex Real-Time Embedded Systems. Licentiate Thesis, Mälardalen University, 2010.
[2] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat et al. The worst-case execution-time problem—overview of methods and survey of tools. ACM Transactions on Embedded Computing Systems, 7(3):36, 2008.
[3] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper. The Mälardalen WCET benchmarks: Past, present and future. In OASIcs-OpenAccess Series in Informatics, vol. 15. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010
[4] M. Marchesotti, M. Migliardi, and R. Podestà. A measurement-based analysis of the responsiveness of the Linux kernel. In 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems, ECBS 2006, pp. 10-pp. IEEE, 2006.
[5] J. Wegener, and F. Mueller. A comparison of static analysis and evolutionary testing for the verification of timing constraints. Real-time systems 21, no. 3 (2001): 241-268.
[6] S. Bygde. Static WCET analysis based on abstract interpretation and counting of elements. PhD diss., Mälardalen University, 2010.
[7] J. Hansen, S. A. Hissam, and G. A. Moreno. Statistical-based WCET estimation and validation. In Proceedings of the 9th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis. 2009.
[8] M. Bohlin, Y. Lu, J. Kraft, P. Kreuger, and T. Nolte. Simulation-based timing analysis of complex real-time systems. In 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 321-328. IEEE, 2009.
[9] S. Samii, S. Rafiliu, P. Eles, and Z. Peng. A simulation methodology for worst-case response time estimation of distributed real-time systems. In Proceedings of the conference on Design, automation and test in Europe, pp. 556-561. ACM, 2008.
[10] Y. Lu, M. Bohlin, J. Kraft, P. Kreuger, T. Nolte, and C. Norström. Approximate timing analysis of complex legacy real-time systems using simulation optimization. In Proceedings of the Work-In-Progress (WIP) session of the 29th IEEE Real-Time Systems Symposium, pp. 29-32. 2008.
[11] Y. Lu, T. Nolte, I. Bate, and C. Norstrom. Timing analyzing for systems with task execution dependencies. In IEEE 34th Annual Computer Software and Applications Conference (COMPSAC), pp. 515-524. IEEE, 2010.
[12] R. S. Sutton, A. G Barto. 1998. Reinforcement learning: An Introduction. Vol. 1. MIT press Cambridge.
[13] M. Saadatmand, and M. Sjodin. Testing of timing properties in real-time systems: Verifying clock constraints. In 20th Asia-Pacific Software Engineering Conference (APSEC), vol. 2, pp. 152-158. IEEE, 2013.
[14] L. I. Kuncheva, Fuzzy classifiers. *Scholarpedia* 3, no. 1 (2008): 2925.
[15] C. Z. Mooney, Monte carlo simulation. Vol. 116. Sage Publications, 1997
[16] J. Kraft, Y. Lu, C. Norström, and A. Wall. A metaheuristic approach for best effort timing analysis targeting complex legacy real-time systems. In *Real-Time and Embedded Technology and Applications Symposium, RTAS'08.* pp. 258-269. IEEE, 2008
[17] Y. Lu, T. Nolte, J. Kraft, and C. Norstrom. A statistical approach to response-time analysis of complex embedded real-time systems. In IEEE 16th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), pp. 153-160. IEEE, 2010