POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

FIGARO: reinForcement learnInG mAnagement acRoss computing cOntinua

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Andrea De Bettin, 10573185
Fabio Lavezzo, 10609185

Advisor:
Prof. Danilo Ardagna

Co-advisors:
Federica Filippini
Hamta Sedghani

Academic year:
2021-2022

Abstract: The steady increase of Artificial Intelligence applications forced to rethink the old paradigm of Cloud computing due to the growth of the data size to be exchanged and, as a consequence, its cost and its latency delay. Edge computing is a recent paradigm that tries to mitigate these issues by relying on resources at the edge of the network. On one hand, this proves to be effective in reducing the quantity of data sent across the network but, on the other, it creates a scattered environment. Optimally assigning computational tasks to available resources becomes the challenge. Starting from a design-time scenario, we developed a novel runtime framework based on Reinforcement Learning: FIGARO (reinForcement learnInG mAnagement acRoss computing cOntinua). The primary objective was to create a fast and reliable decision tool that could cope with the major challenges involved in a runtime scenario (e.g., variability of the incoming load, network performance variation, changes in end-users' behaviour). To decrease the learning time of the Agent, design-time knowledge has been exploited throughout its training process. The experimental results highlight how we achieved a speed-up of about 10 times with respect to our baseline software. Our Static Reinforcement Learning (RL) Agent proved its capability to match the performance of the benchmark under various environment assumptions. Moving to the Dynamic RL Agent, featuring continuous learning, the performance overcomes the benchmark even in a short episode when the response times follow a different distribution with respect to the one used during the preliminary offline training.

Key-words: Edge computing, Reinforcement Learning, Deep Q-Learning, Optimal allocation

1. Introduction

One of the main reasons behind the rise of the Cloud computing paradigm is the fact that it makes an ideally unlimited computational and storage power accessible according to pay-to-go pricing models. The Cloud market is seemingly experiencing a continuous growth and experts expect its size to reach 1600 USD billion by 2030 [1]. In the last few years, an accelerated migration towards mobile computing and Internet of Things (IoT) is designing a new framework, where high benefits can be achieved by exploiting resources at the edge of the network. Consequently, Edge computing was proposed as an alternative to Cloud.

The Edge computing paradigm brings storage, computation and network services near the end devices through the so called fog nodes, which can be switches, cameras, routers, computers and servers, linked through a network connection [2].
Thanks to this approach, Edge computing is characterised by low latency, processing power closer to the user and real time interactions, overcoming Cloud computing limitations [3].
Nonetheless, it should be noticed that Edge resources have usually less computing capacity than the Cloud, and can become a bottleneck in the computation. Hence, if an application requires computational power which is not available at the Edge nodes, there should still be the option to connect to the Cloud data centers [2], if latency requirements and network connections allow so.

Therefore, the *Computing Continuum* paradigm is the best approach: latency-sensitive tasks can be distributed among Edge nodes while compute-intensive tasks are offloaded to the Cloud layers.
Since in this scenario the computation is distributed among devices with highly heterogeneous capacities and connectivity, efficient resource allocation and component placement algorithms are crucial to orchestrate at best the physical resources of the Computing Continuum, minimising the expected costs while meeting constraints related to, e.g., Quality of Service (QoS) response times.

In many literature proposals (e.g., [4]), the problem of component placement and resource selection for Artificial Intelligence (AI) applications in the Computing Continuum is faced at design time, dealing with different application requirements.
The tool that is presented in [4], which has been used as starting point in this work and is referred to in the following as Space4AI-D, is based on a randomised greedy algorithm whose objective is to determine the optimal allocation of the application components on the candidate resources of minimum cost, providing performance guarantees across heterogeneous resources of different types (e.g., Edge devices, Cloud GPU-based Virtual Machines (VMs) and Function as a Service solutions (Faas)).
Despite the very good quality of Space4AI-D solutions, in practical applications the workload (e.g., requests per second to be processed) expected at design-time is often subject to fluctuations due, for example, to variations in the generated data volumes. For this reason, the initially designed placement may become unfeasible or oversized depending on the input workload, so it has to be continuously monitored and adapted online.

Starting from the Space4AI-D design-time tool, we developed an integrated framework that handles the runtime. A runtime tool must provide a feasible reconfiguration in few seconds at most, due to the online nature of the running applications. Most of them are not indeed long tasks, for example, they may revolve around an evaluation of a Neural Network. Therefore, runtime tools must be designed to make appropriate cost-reactivity trade-offs while tuning the solver algorithms, as well as implemented in a very efficient way to reduce the execution time.

To tackle the runtime problem, we opted for the great possibilities offered by Reinforcement Learning (RL). Other works, such as [5], have already highlighted how a RL methodology can successfully adapt overtime to variations in, e.g., the workload model, by relying only on observations extracted from the environment. Therefore, a RL-based approach can deal with shifting aspects of AI applications (e.g., variability of the incoming load, network performance variation) and of the the end-users' behaviour. It is also more flexible with respect to resource management methods based on analytical models and an optimisation problem.
The downside of RL is the huge time needed by the Agent to learn an effective policy. During the training the Agent needs to explore all the actions it may select and some undesired behaviours may happen, e.g., leading to violations of the QoS constraints.
The main contribution of this work is to develop FIGARO, a runtime framework where the power of the RL-based methods is merged with the knowledge (i.e., the solutions) of a design-time approach, aiming to get the best out of both worlds.
The FIGARO environment embeds:

- A Simulator to compute the response times without imposing the strict hypotheses that an analytical approach would. An example is the possibility to vary the distribution used to draw service times.
- A RL-based Agent able to take fast decisions. It capitalises on both an initial share of knowledge coming from a design-time analysis and every information acquired while running.
- A design-time tool, Space4AI-D, serving multiple purposes. It is used to compute the response times and to represent the analytical counterpart of the Simulator, allowing a comparison. But it can also behave as the Agent deployed inside the environment, becoming a reference solution to assess the RL-based Agent.

We would like to stress further how the knowledge of Space4AI-D is employed to get an initial approximated policy, that is, a policy mimicking the Space4AI-D design-time behaviour. On one hand, this improves the learning procedure of the Agent and, on the other hand, it prevents a great deal of QoS violations because the Agent immediately deploys an effective policy instead of starting with a completely random one.

Figure 1 shows the cumulative violation rate obtained by two Static RL-based Agents. The first (red line) has just started its preliminary training whilst the second (black line) has already achieved a performance comparable to Space4AI-D by resembling its behaviour. The latter accomplishes better performance in terms of both number and severity of QoS violations. Fully exploiting the design-time knowledge leads to an already effective initial policy for the RL Agent.

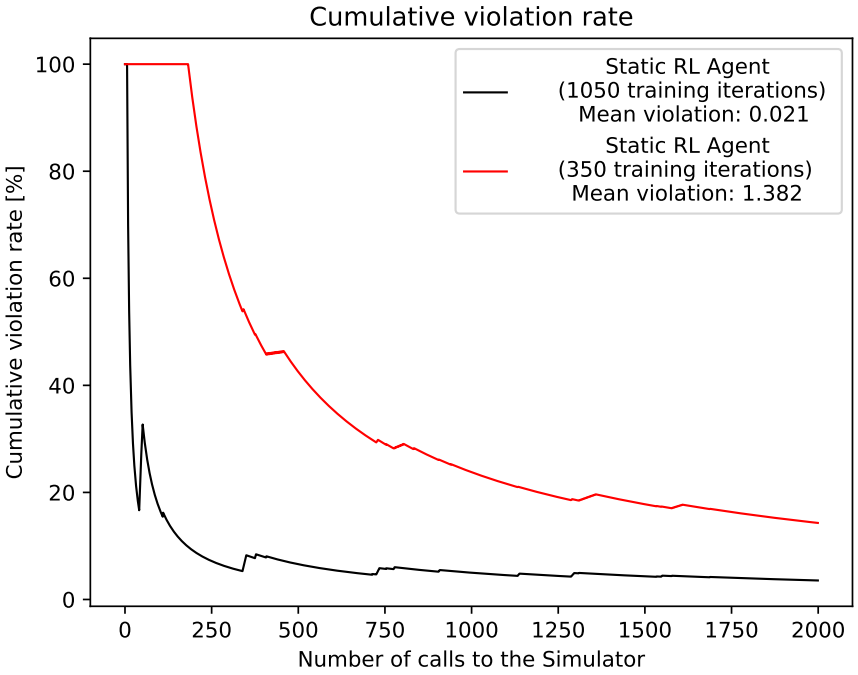
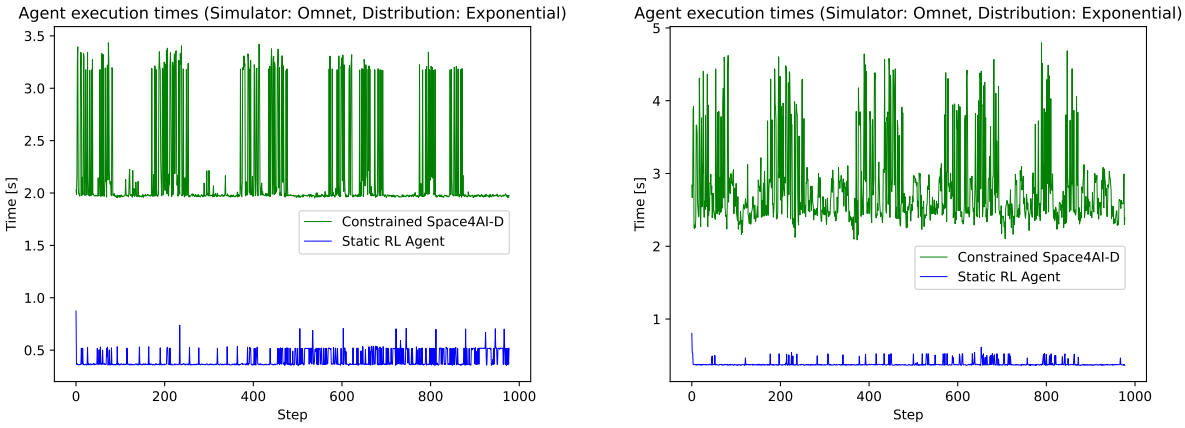


Figure 1: Comparison of the cumulative rate of QoS violations of the two Static RL Agents trained with different number of training iterations - one-component system.

This thesis is organised as follows. Related works are discussed in Section 2. Section 3 introduces the basic application and resources models. Section 4 formulates the optimisation problem, while Section 5 focuses on the RL part. Section 6 presents the FIGARO framework. Experimental results are discussed in Section 7, and final conclusions are drawn in Section 8.

2. Related work

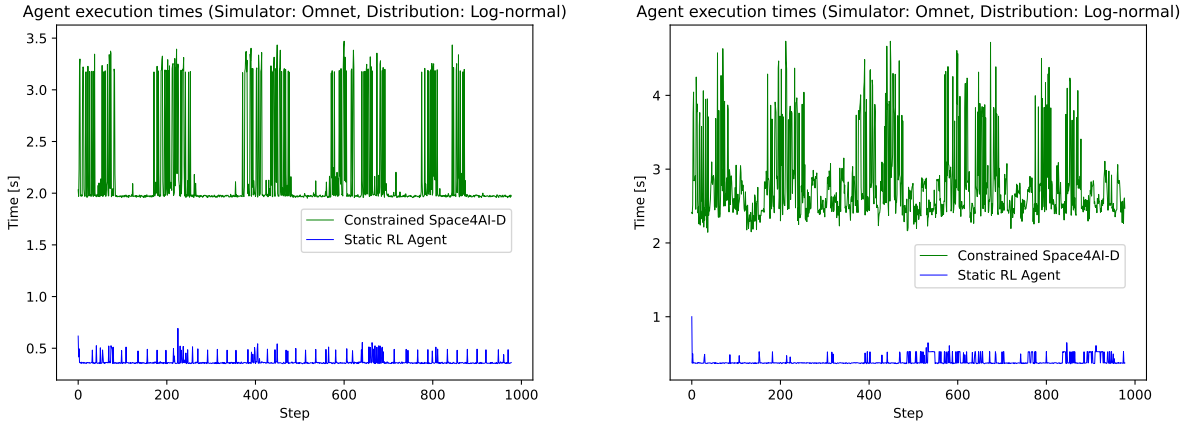
Edge computing, Cloud computing, Fog computing and the possible applications of Reinforcement Learning are recurrent themes in the latest research trends.
The authors in [6] tackle an offloading problem via an Unmanned Aerial Vehicles (UAV) used as Edge server. A helper UAV is deployed to both assist the legitimised UAV and to contrast an eavesdropper UAV (i.e., switching between a *relay* mode and a *jammer* mode). The choice of this mode, as well as other parameters such as the velocity of the helper UAV, is determined by adopting a Deep Deterministic Policy Gradient based method, a type of Deep Reinforcement Learning suited for continuous control problems.
Offloading a task to the nearest Mobile Edge Computing (MEC) server may not be the optimal solution due to its limited computing resources. Jointly optimising the offloading decision, namely where to offload a task, and resource management, namely how much computing resource in an MEC server is allocated to a task, is critical. In [7] the authors model this optimisation problem as Markov Decision Process (MDP) and propose the Deep reinforcement lEarning based offloading deCision and rEsource managemeNT (DECENT) algorithm, which leverages the Advantage Actor Critic method to optimise the offloading decision and computing resource allocation for each arriving task in real-time such that the cumulative weighted response time can be minimised.
The theme of workflow offloading continues in [8] to face the problems related to the massive network traffic and calculation workload among end-users and Cloud platforms. The authors address the robustness of the offloading policies from a multi-objective perspective by proposing a Meta-Reinforcement-Learning-based



(a) One-component system.

(b) Two-components system.

Figure 28: Execution times of the Agents - exponential service times.



(a) One-component system.

(b) Two-components system.

Figure 29: Execution times of the Agents - log-normal service times.

8. Conclusions

The importance of Edge computing has dramatically risen in the recent period. As showed in the Section 2, an active community of researchers works to find new algorithms to fully harness the potential of Edge computing. The main goal of such works is the optimisation of resource usage to achieve better performance and reduce the costs, the time spent and the consumed energy. Our work was aligned with this effort. Starting from a design-time framework, we moved towards a runtime framework in order to achieve adaptability with respect to changes in the environment and to abandon some restrictive assumptions, such as exponentially distributed service times. The proposed solution is a Reinforcement Learning (RL) Agent implementing the Deep Q-Learning algorithm. A key element is exploiting the design-time knowledge to speed up the learning process of the RL and to deploy an already effective policy from the very beginning. We achieved a reduction of computational times, constant regardless of the system dimensionality, of about 10 times with respect to the benchmark (Space4AI-D). Shifting the environment assumptions, the Static RL Agent showed behaviours similar to Space4AI-D. Moving to the Dynamic RL Agent, featuring continuous learning, the performance overcomes the benchmark even in a short episode when the response times follow a different distribution with respect to the one used during the preliminary training.

References

[1] Precedence Research. Cloud computing market size to hit us $ 1,614.1 billion by 2030. URL "https://www.globenewswire.com/en/news-release/2022/05/13/2443081/0/en/Cloud-ComputingMarket-Size-to-Hit-US-1-614-1-Billion-by-2030.html".

[2] Sabireen H. and Neelanarayanan V. A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges. *ICT Express*, 7(2):162–176, 2021. ISSN 2405-9595. doi: https://doi.org/10.1016/j.icte.2021.05.004. URL https://www.sciencedirect.com/science/article/pii/S2405959521000606.

[3] Michaela Iorga, Larry Feldman, Robert Barton, Michael Martin, Nedim Goren, and Charif Mahmoudi. Fog Computing Conceptual Model, 2018-03-14 2018.

[4] Hamta Sedghani, Federica Filippini, and Danilo Ardagna. A Random Greedy based Design Time Tool for AI Applications Component Placement and Resource Selection in Computing Continua. In *2021 IEEE International Conference on Edge Computing (EDGE)*, pages 32–40, 2021. doi: 10.1109/EDGE53862.2021.00014.

[5] Enda Barrett, Enda Howley, and Jim Duggan. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, 25(12):1656–1674, 2013. doi: https://doi.org/10.1002/cpe.2864. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.2864.

[6] Seonghoon Yoo, Seongah Jeong, and Joonhyuk Kang. Hybrid uav-enabled secure offloading via deep reinforcement learning, 2022. URL https://arxiv.org/abs/2208.07550.

[7] Zeinab Akhavan, Mona Esmaeili, Babak Badnava, Mohammad Yousefi, Xiang Sun, Michael Devetsikiotis, and Payman Zarkesh-Ha. Deep reinforcement learning for online latency aware workload offloading in mobile edge computing, 2022. URL https://arxiv.org/abs/2209.05191.

[8] Hongyun Liu, Ruyue Xin, Peng Chen, and Zhiming Zhao. Multi-objective robust workflow offloading in edge-to-cloud continuum. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, pages 469–478, 2022. doi: 10.1109/CLOUD55607.2022.00070.

[9] Somayeh Yeganeh, Amin Babazadeh Sangar, and Sadoon Azizi. A novel q-learning-based hybrid algorithm for the optimal offloading and scheduling in mobile edge computing environments. *Journal of Network and Computer Applications*, 214:103617, 2023. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2023.103617. URL https://www.sciencedirect.com/science/article/pii/S108480452300036X.

[10] Hongyun Liu, Peng Chen, and Zhiming Zhao. Towards a robust meta-reinforcement learning-based scheduling framework for time critical tasks in cloud environments. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 637–647, 2021. doi: 10.1109/CLOUD53861.2021.00082.

[11] Yihong Li, Xiaoxi Zhang, Tianyu Zeng, Jingpu Duan, Chuan Wu, Di Wu, and Xu Chen. Task placement and resource allocation for edge machine learning: A gnn-based multi-agent reinforcement learning paradigm, 2023. URL https://arxiv.org/abs/2302.00571.

[12] Malathy Navaneetha Krishnan and Revathi Thiyagarajan. Multi-objective task scheduling in fog computing using improved gaining sharing knowledge based algorithm. *Concurrency and Computation: Practice and Experience*, 34(24):e7227, 2022. doi: https://doi.org/10.1002/cpe.7227. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.7227.

[13] Yaohua Sun, Jianmin Chen, Zeyu Wang, Mugen Peng, and Shiwen Mao. Enabling mobile virtual reality with open 5g, fog computing and reinforcement learning. *IEEE Network*, pages 1–18, 2022. doi: 10.1109/MNET.010.2100481.

[14] Bao Trinh and Gabriel-Miro Muntean. A deep reinforcement learning-based offloading scheme for multi-access edge computing-supported extended reality systems. *IEEE Transactions on Vehicular Technology*, pages 1–10, 2022. doi: 10.1109/TVT.2022.3207692.

[15] Nam H. Chu, Diep N. Nguyen, Dinh Thai Hoang, Khoa T. Phan, Eryk Dutkiewicz, Dusit Niyato, and Tao Shu. Dynamic resource allocation for metaverse applications with deep reinforcement learning, 2023.

[16] Xin Peng, Zhengke Han, Wenwu Xie, Chao Yu, Peng Zhu, Jian Xiao, and Jinxia Yang. Deep reinforcement learning for shared offloading strategy in vehicle edge computing. *IEEE Systems Journal*, pages 1–12, 2022. doi: 10.1109/JSYST.2022.3190926.

[17] Philipp Raith, Thomas Rausch, Schahram Dustdar, Fabiana Rossi, Valeria Cardellini, and Rajiv Ranjan. Mobility-aware serverless function adaptations across the edge-cloud continuum.

[18] Guangyao Zhou, Ruiming Wen, Wenhong Tian, and Rajkumar Buyya. Deep reinforcement learning-based algorithms selectors for the resource scheduling in hierarchical cloud computing. *Journal of Network and Computer Applications*, 208:103520, 2022. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2022.103520. URL `https://www.sciencedirect.com/science/article/pii/S1084804522001618`.

[19] Wenting Wei, Huaxi Gu, Kun Wang, Jianjia Li, Xuan Zhang, and Ning Wang. Multi-dimensional resource allocation in distributed data centers using deep reinforcement learning. *IEEE Transactions on Network and Service Management*, pages 1–1, 2022. doi: 10.1109/TNSM.2022.3213575.

[20] Fei Xu, Jianian Xu, Jiabin Chen, Li Chen, Ruitao Shang, Zhi Zhou, and Fangming Liu. igniter: Interference-aware gpu resource provisioning for predictable dnn inference in the cloud, 2022. URL `https://arxiv.org/abs/2211.01713`.

[21] Z. Chen, B. Zhu, and C. Zhou. Container cluster placement in edge computing based on reinforcement learning incorporating graph convolutional networks scheme. *Digital Communications and Networks*, 2023. doi: https://doi.org/10.1016/j.dcan.2023.02.012.

[22] Martin Straesser, Simon Eismann, Jóakim von Kistowski, André Bauer, and Samuel Kounev. Autoscaler evaluation and configuration: A practitioner's guideline. In *Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering*, ICPE '23, New York, NY, USA, 2023. Association for Computing Machinery. In print.

[23] Danilo Ardagna and Barbara Pernici. Adaptive Service Composition in Flexible Processes. *IEEE Transactions on Software Engineering*, 33(6):369–384, 2007. doi: 10.1109/TSE.2007.1011.

[24] Lizheng Jiang, Yunman Pei, and Jiantao Zhao. Overview Of Serverless Architecture Research. *Journal of Physics: Conference Series*, 1453(1):012119, jan 2020. doi: 10.1088/1742-6596/1453/1/012119. URL `https://dx.doi.org/10.1088/1742-6596/1453/1/012119`.

[25] Amazon. AWS Lambda: Run code without thinking about servers or clusters, 2022. URL `https://aws.amazon.com/lambda/`.

[26] Amazon. AWS Lambda Pricing, 2022. URL `https://aws.amazon.com/lambda/pricing/`.

[27] Microsoft. Azure Functions pricing, 2022. URL `https://azure.microsoft.com/en-us/pricing/details/functions/`.

[28] Amazon. AWS Step Function pricing, 2022. URL `https://aws.amazon.com/step-functions/pricing/`.

[29] Microsoft. Azure Logic Apps, 2022. URL `https://azure.microsoft.com/en-us/products/logic-apps/`.

[30] Sebastián Risco, Germán Moltó, Diana M Naranjo, and Ignacio Blanquer. Serverless Workflows for Containerised Applications in the Cloud Continuum. *Journal of Grid Computing*, 19, 2021.

[31] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. ISBN 0262039249.

[32] Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow. `https://github.com/tensorflow/agents`, 2018. URL `https://github.com/tensorflow/agents`. [Online; accessed 25-June-2019].

[33] TensorFlow. TensorFlow-Agents documentation, 2023. URL `https://www.tensorflow.org/agents`.

[34] TensorFlow. TensorFlow repository, 2023. URL `https://github.com/tensorflow/tensorflow`.

[35] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow:: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[36] Bergstra J. Yamins D. Cox D. D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. TProc. of the 30th International Conference on Machine Learning , 2013. URL `https://hyperopt.github.io/hyperopt/`.

[37] Bergstra J. Yamins D. Cox D. D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. TProc. of the 30th International Conference on Machine Learning (ICML 2013), 2013. URL `https://github.com/hyperopt/hyperopt`.

[38] OpenSim Ltd. Omnet++ documentation, 2023. URL `"https://omnetpp.org"`.

[39] OpenSim Ltd. Omnet++ repository, 2023. URL `"https://github.com/omnetpp/omnetpp"`.

Abstract in lingua italiana

La crescente importanza di applicazioni basate sull'Intelligenza Artificiale ha condotto a dei ripensamenti riguardo al paradigma Cloud Computing, specialmente a causa delle quantità sempre maggiori di dati che devono essere scambiati e che portano a costi e ritardi legati alla comunicazione. Edge Computing è una proposta recente che si prefigge di mitigare alcuni di questi problemi tramite l'utilizzo di tutte le risorse disponibili, anche quelle al margine della rete. Se da un lato è capace di ridurre la quantità di dati trasmessi, dall'altro conduce alla frammentazione del sistema. Pertanto, assegnare in modo ottimale i vari processi alle risorse disponibili risulta cruciale. Partendo da alcune analisi effettuate in fase di definizione del problema (EN: at design-time), abbiamo lavorato per giungere a una visione del problema durante la sua esecuzione (EN: at run-time) creando una struttura decisionale basata sull'Apprendimento per Rinforzo (EN: Reinforcement Learning) che chiamiamo FIGARO: reinForcement learnInG mAnagement acRoss computing cOntinua. L' obiettivo principale è stato creare uno strumento veloce e affidabile che potesse affrontare le sfide che emergono durante le fasi di esecuzione (ad esempio, carichi di lavoro variabili, modifiche nel comportamento degli utenti). Per ridurre il tempo di apprendimento dell'Agente, abbiamo sfruttato la conoscenza derivante da metodi concepiti in fase di definizione durante la procedura di addestramento. I risultati sperimentali mostrano come il nostro Agente abbia dei tempi di esecuzione circa 10 volte inferiori rispetto al programma utilizzato come riferimento. L'Agente Statico riesce ad avere prestazioni paragonabili a quelle del riferimento. Utilizzando l'Agente Dinamico, che continua l'apprendimento durante il suo impiego, il rendimento supera quello del modello confrontato, nonostante la lunghezza limitata dell'episodio e la differente distribuzione dei tempi di risposta impiegata durante l'addestramento iniziale.

Parole chiave: Edge computing, Apprendimento per Rinforzo, Deep Q-Learning, Allocazione ottimale

Acknowledgements

We would like to thank our supervisor, Danilo Ardagna, for his guidance and patience throughout our experience together. Super special thanks to Federica Filippini who helped us during every stage of our work, including also the tricky technical parts. We are also grateful to Hamta Sedghani for her help with Space4AI-D.
We thank Riccardo Lancellotti, professor at the Università degli Studi di Modena e Reggio Emilia, for his invaluable support when dealing with the Simulator. We guess we tested his patience enough.
Finally, we want to thank our university mates that shared with us triumphs and downfalls by studying together in the mythical *Interfacoltà*.