NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium | 978-1-6654-0601-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NOMS54207.2022.9789757

Application Placement in Fog Environments using Multi-Objective Reinforcement Learning with Maximum Reward Formulation

Reinout Eyckerman, Phil Reiter,
Steven Latré, Johann Marquez-Barja and Peter Hellinckx
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Sint-Pietersvliet 7, 2000 Antwerp, Belgium
Email: {firstname.lastname}@uantwerpen.be

***Abstract*—The service placement problem considers the placement of multiple connected services across a heterogeneous device network and is one of the core problems of fog computing. We discuss the complexity of this service placement problem, and propose a model for solving it using Multi-Objective Reinforcement Learning (MORL) methodologies. Using a trained neural network greatly reduces the resource consumption of the placement algorithm, making it viable for resource-constrained scenarios. Starting from state-of-the-art techniques, we develop a generic max reward formulation model and apply several MORL methodologies, which solve the placement problem in scenarios where the preference weights change. We compare the results to a baseline methodology and showcase the value of MORL on the placement problem.**

I. Introduction

There will be an estimated 29.3 billion networked devices by 2023 [1]. Based on current computing standards, most of the information generated by these devices will be processed by a remote cloud server. However, cloud computing is not properly suited to support low-latency applications, as the distance between cloud and end devices can have a detrimental impact on the response time. In addition, large numbers of devices have a sizeable impact on the network as these devices tend to collectively generate large amounts of data. This problem is partly covered by the upcoming 5G networks, which aim to support ultra-reliable, low latency applications and high throughput Internet of Things (IoT) devices [2].
However, 5G networks only solve a part of the problem, as applications and changing environments will still incur a bandwidth and latency impact causing network degradation. To this end, fog computing is proposed, a paradigm which encourages using the resources available closer to the end user [3]. An example of a fog network is showcased in Fig. 1. This is similar to edge computing, where additional compute resources are added at key locations closer to the end user. Fog computing is designed to support applications using IoT devices, by offloading services to nearby devices to support more compute-intensive, latency-aware, and even energy-aware applications, while suppressing the latency impact caused by cloud communication while not overloading the network. Providing these services in the fog requires effective service placement, as placing them far away from related services can induce latency costs, and placing all the services on a single device will consume more device resources than are available, reducing service reliability.

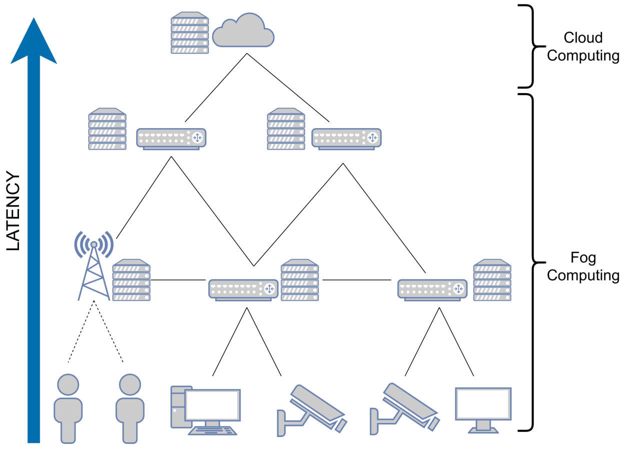

Fig. 1. An Example of a Fog Network

To this service allocation problem, we propose a general Deep Q-Network (DQN) methodology for service placement, which considers the placement optimization of multiple intercommunicating services on a network. This placement optimization considers the improvement of the network performance by minimizing the total impact on the network, while also improving energy efficiency. These are generally conflicting objectives, defining the problem as a Multi-Objective Optimization (MOO) problem. Valid placements should satisfy device con-

This research received funding from the Flemish Government (AI Research Program). This article describes work in the context of the DEDICAT 6G project under the European Union (EU) H2020 research and innovation programme (Grant Agreement No. 101016499). The contents of this publication are the sole responsibility of the authors and do not in any way reflect the views of the EU.

978-1-6654-0601-7/22/$31.00 © 2022 IEEE

straints, such as available memory, network constraints, such as available bandwidth, and application constraints, such as maximally allowed latency between two services. In this paper, we applied several State-of-the-Art (SotA) Multi-Objective Reinforcement Learning (MORL) methods to solve the service allocation problem, using the strengths of DQNs to effectively place the services across the network. Our proposed methodologies use scalarization and support dynamic weight changes throughout the network lifespan, providing support for a higher-level control mechanism.

II. Related Work

A. Service Allocation Approach

The field of service allocation is already widely researched, and the research often considers different objectives, constraints and requirements. This allows for a large variety of methodologies, each with their own strengths and weaknesses. However, many methodologies depend on reductions which only hold when considering very specific use-cases. Salaht et al. [4] provide an extensive overview of the Service Allocation Problem in the context of fog and edge computing. Some interesting findings include that most of the research in this field only considers a single objective problem, focused on latency minimization. This single-objective research often contains a weighted sum of multiple different objectives, but often fails to mention that this is fundamentally a multi-objective problem on which reward shaping was applied. The most applied solving techniques use either a variant of Integer Programming or Constrained Optimization. The survey showcases that most researchers focus on a single use-case, shown in the large variety of validation simulation environments and test-beds. We derive that this focus on a specific use-case causes that all these papers use their own specific set of objectives and constraints, making it exceptionally difficult to compare results. This is showcased throughout most research papers in the field, which usually only compare to a baseline approach and a naive greedy implementation.

Tang et al. propose using a DQN for service placement and migration in fog networks and reduce the problem to a bin-packing problem [5]. The goal is to minimize power consumption, communication delay and migration cost, while adhering to the resource constraints. They consider full control of the network, as nodes that do not run any services get shut down to conserve energy. The constraints mentioned are to be modeled in the action space of the exploration phase, so that actions being constrained do not occur and are excluded from the valid action space, but are further not mentioned. In addition, an improved action selection policy is proposed, where services running on overloaded devices are preferred to be moved to devices with more available resources, while devices with very few services also move their services to another device to conserve energy. They compare their adapted Q-learner to regular Q-learning methodologies and constraint-conserving heuristics, and showcase that the improvements to the Q-learning algorithm allow it to outperform the baselines. Gazori et al. propose a Double Deep Q-Network (DDQN) approach for service placement in the fog [6]. They compare several existing service scheduling approaches and reduce the problem to a tree graph. The optimization goal is to minimize the End-to-End (E2E) service delay and computation cost, under deadline and resource constraints. They show the impact of the hyperparameters on the agent behaviour, and compare the DDQN to a first-fit, greedy, random and basic tabular Q-learner. The proposed scenarios were slightly resource constrained, and, due to limited training diversity, the agents were prone to overfitting. The results show that the DDQN approach outperforms the other algorithms on all metrics. Mseddi et al. tackle the user request satisfaction problem within fog environments, handling it as a more complex version of the Generalized Assignment Problem (GAP) [7]. This is done by tackling the user requests individually and using the Reinforcement Learning (RL) agent to schedule the requested service individually. This approach supports dynamic scenarios but increases the difficulty of finding a globally optimal goal. The approach is evaluated against an Integer Linear Programming (ILP) technique, nearest allocation and random allocation methodology and shown to outperform these methodologies.

B. Problem Definition

Various research in the SotA defines the placement problem differently, although they tackle similar problems. We now showcase how other research defines these placement problems, and show how our problem statement differs. iFogStor defines a generic model for application placement using the GAP problem formulation [8]. GAP represents the problem where M different items are to be distributed across N different bins, while ensuring that the size of all items in a specific bin does not exceed the bin capacity, to maximize the value $f(i,j), i \in M, j \in N$ of all items across all bins. This problem model is useful for optimizing towards device specific constraints and objectives, such as load balancing and energy consumption, but does not incorporate any network behavior. Gu et al. [9] showcase the similarity of the placement problem with the General Quadratic Assignment Problem (GQAP), which maps n different equipment across m different locations, while considering the distance between equipment and capacity constraints on the locations. This problem definition enables the modeling of the impact on the network, and will consequently be used as the basis for the rest of this paper.

III. Service Allocation Problem

A. Problem Setting

The service allocation problem is a generalization on the previously defined GQAP. The service allocation problem expands on this by putting constraints on both the devices and the network, whereas the GQAP does not consider transport constraints. GQAP has been shown to be solvable for up to 22 devices, showcasing the problem complexity [10]. Additional complexities arise due to applying it to fog computing: the available run-time of the placement algorithm is generally very low, as it has seconds to find a solution, depending on the context. Similarly, the available resources of fog devices are

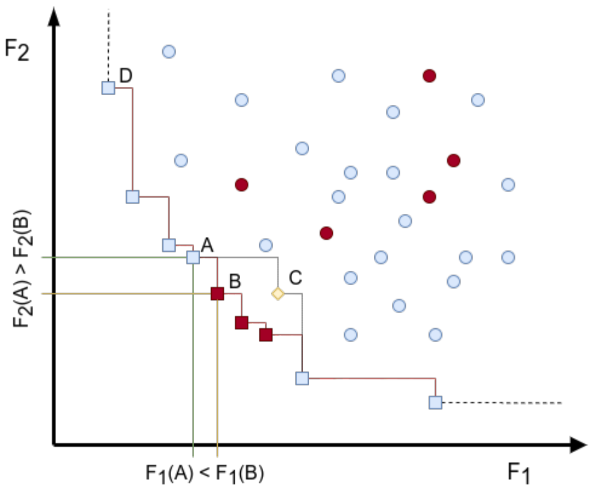

Fig. 2. An example discrete Pareto Front. The square nodes represent the Pareto-dominant solutions, where the circles represent the dominated solutions. The diamond shape, C, shows how the Pareto Front would change if constrained solutions, shown in dark red, are removed from the Pareto Front.

generally limited, requiring algorithms with low computational complexity. This further bounds the methodologies applicable to the problem. We additionally extend the GQAP to consider MOO, as the service allocation problem considers multiple competing objectives, defined further below.

B. Multi-Objective Optimization

MOO considers the optimization of multiple, possibly conflicting, objectives. This is formulated in Equation 1 [11].

$$\begin{aligned} \min\ & \mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}), F_2(\mathbf{x}), \ldots, F_o(\mathbf{x})] \\ \text{s.t.}\ & \mathbf{x} \in X \\ & g_i(\mathbf{x}) \geq 0, i = 1, 2, \ldots, k \\ & h_j(\mathbf{x}) = 0, j = 1, 2, \ldots, l \end{aligned} \tag{1}$$

The goal is to traverse the search space $\mathbf{x} \in X$ to simultaneously minimize the cost of the objective function vector $\mathbf{F}$ for o different objectives, while considering k different inequality constraints, g_i, and l different equality constraints, h_j.
These objectives conflict, thus there is no single optimal solution, rather a front of non-dominated solutions. This front is the Pareto Front (PF), where each solution is at least better for one objective while being worse for at least one other objective. This is shown in Fig. 2, where a bi-objective PF is shown. Solutions A and B are non-dominated solutions, signifying that they are worse on at least one objective while being better on at least one other objective. Both solutions dominate C, which is equal to B concerning objective F_2, but worse for objective F_1. However, the search space for the placement problem is constrained. The solutions which do not satisfy the constraints are shown on Fig. 2 as dark red nodes. By removing these from the search space, it becomes clear that solution C becomes part of the PF, as B becomes infeasible. However, the entire PF is not necessary for the service allocation problem, only one solution is required from this front. The selection of this solution is done by a Decision Maker (DM), which defines the preferences for the set of objectives. This definition is often done using scalarization.
By summing the objectives together, the utility of a solution can be defined as a linear combination of the objectives. This scalarization is done using a weighted function, where the weights represent the importance of each objective. Popular techniques for this approach include the weighted sum approach, which purely uses the weights defined by the DM, and the Chebychev scalarization, which also incorporates a reference vector, giving a search direction to the algorithm [12]. Scalarization is one of the simplest approaches for solving MOO problems, but it is often difficult for the DM to find the optimal set of weights. This has an especially large impact on Machine Learning (ML) scenarios, where the training of weights for a neural network can consume a large amount of time and resources.

IV. Reinforcement Learning Approach

A. Markov Decision Process

General RL problems are represented as Markov Decision Processes (MDPs), which ensure convergence towards an optimal policy. As this research is MOO oriented, we consider a Multi-Objective Markov Decision Process (MOMDP), an extension of the existing MDP. A MOMDP can be represented as a 6-tuple $(S, A, P, \mathbf{r}, \omega, f_\Omega)$, with S representing the state space, A the action space, $P(s'|s, a)$ representing the transition distribution, $\mathbf{r}(s, a)$ is the vector reward function, Ω is the space of preference, and f_Ω is preference function, which takes preference $\omega \in \Omega$ as inputs and outputs the scalar utility for each action in a given state, i.e., $f_\omega(\mathbf{r}(s, a)) = \omega\mathbf{r}(s, a)$ [13]. Our goal is to maximize the utility function $u(\omega)$, representing the scalarization of the problem for any preference vector $\omega \in \Omega$ defined at run-time.

B. Multi-Objective Reward

In this paper, we propose using a centralized, offline, dynamic RL algorithm to solve the service placement problem. This centralized approach enables global optimization through a full view of the network state and reduces the data sharing complexity. The field of MOO is gaining traction in the RL community. Rădulescu et al. provide us with an expansive survey on scalarized multi-objective and multi-agent RL [14]. One core component they showcase is the difference between Scalarized Expected Returns (SER) and Expected Scalarized Returns (ESR) in scalarized MORL. They showcase the degree of freedom in which the utility function can be integrated into the reward function. In a SER-oriented approach, scalarization first computes the payoffs of the policy and then applies the scalarization methodology:

$$\mathbf{V}_u^\pi = u\left(E\left[\sum_{t=0}^{\infty} \gamma^t \mathbf{r}_t | \pi, \mu_0\right]\right) \tag{2}$$

This is the correct approach when the policy is executed multiple times, and where the average execution over multiple policies is important to the agent. In a ESR-oriented approach, scalarization first applies the scalarization methodology before computing the payoffs:

$$\mathbf{V}_u^\pi = E\left[u\left(\sum_{t=0}^{\infty}\gamma^t \mathbf{r}_t\right)|\pi,\mu_0\right] \tag{3}$$

This methodology is used when the return of the policy over a single run is important for the agent. The core focus of the proposed approaches is scalarized MORL. A naive approach of solving the placement problem is to predefine a set of weights for which it will be solved, and apply that solution. During subsequent runs, different weights can be used to represent changing preferences. This methodology works well for generic Evolutionary Algorithms (EAs) and heuristics, but is less applicable to ML methodologies due to the complexity and long training times. In this regard, the focus is on leveraging RL methodologies which consider these dynamic weights.

V. Methodology

A. Problem Model

We now define the MDP model.

1) Observation Space: The problem has been modeled as a fully observable MOMDP. The observation space is the allocation array, which represents the current mapping of the services to the devices.

2) Action Space: The agent takes appropriate actions by moving a single service to specific device, manipulating the allocation array. The agent gets a penalty if it takes an action which would violate the device constraints, promoting finding valid placements. Although a dynamic action space through masking would be useful to prevent taking actions which lead to a constrained solution, a static approach was selected. This approach allows our agent to traverse through constrained parts of the search space, which is especially important in scenarios with a low degree of unconstrained solutions.

3) Reward: The agent is rewarded on both the level of constraint satisfaction and the value of each objective. To support constrained environments, the objectives have been given a penalty based on how many constraints are exceeded, as defined in Eq. 4.

$$C_p = \sum_{r}^{|R|}(C_r) \tag{4}$$

The parameter $C_r \in \{0,1\}$ represents 1 if constraint r is satisfied, 0 otherwise, and scales the impact of the constraints in case the objectives are not satisfied. All objectives are penalized equally. The entire approach to find the reward vector is defined in Eq. 5.

$$\mathbf{r}(s_t,a_t) = [(F_1(s_t)+C_p),(F_2(s_t)+C_p),\ldots,(F_n(s_t)+C_p)] \tag{5}$$

The reward vector $\mathbf{r}$ is constructed by taking the objective value $F_i(s_t), i \in n$ for n different objectives. By separating the constraints, we ensure their impact on the overall learning, regardless of the current weight. By modeling the constraints and the objectives directly into the reward, we formulate a problem where the agent should find an optimal state, as the value of the state is what matters in this problem, not the trajectory to getting in a specific state. However, general RL optimizes the expected cumulative return, which does apply value to the trajectory. We resolve this issue by formulating the problem as a max reward problem, where the expected maximum reward along a trajectory should be optimized. This approach was proposed by Gottipati et al., who showcased its behavior and gave proof of convergence [15]. The proposed Bellman equation is shown below:

$$Q_{\max}^\pi(s_t,a_t) = \tag{6}$$

$$\mathbb{E}_{\substack{s_{t+1}\sim P(\cdot|s_t,a_t)\\ a_{t+1}\sim\pi(\cdot|s_{t+1})}}\left[\max\left(r(s_t,a_t),\gamma Q_{\max}^\pi(s_{t+1},a_{t+1})\right)\right] \tag{7}$$

Here, the Q value of the policy π is maximized, by taking the max value between the current reward and the expected Q value of the next state. This approach encourages the model to search for maximized reward values, instead of the discounted return. Combined with the maximum reward formulation defined in Eq. 7, this becomes the following:

$$\mathbf{Q}_{u,max}^\pi = \mathbb{E}\left[u\left(\underset{t\to\infty}{max}(\mathbf{r}_t,\gamma\mathbf{Q}_{max}^\pi(s_{t+1},a_{t+1}))\right)|\pi,\mu_0\right] \tag{8}$$

The utility function u is a weighted sum, shown in Eq. 9.

$$u(r) = \sum_{i=1}^{n} r_i * \omega_i \tag{9}$$

4) Terminal Condition: The approach proposed for solving the MOMDP does not allow for any terminal conditions, since the agent will never be certain to know when it has reached a sufficient maximum. To this end, we let the agent traverse through its entire episode and use the maximum reward formulation, as defined in Eq. 7.

B. Linear Scalarization

Linear scalarization is the baseline approach for solving generic MOO problems. It uses the strength through simplicity approach of the weighted sum, where objectives are multiplied with a predefined weight and summed together, as defined in Eq. 9. This approach has been applied in the SotA already, including Tang et al. [5]. While this approach enables tackling MOO problems with single-objective approaches, it suffers from the inability to handle a change in the weights of the objectives. To this end, we propose expanding on the scalarization approach by intelligently managing the weights of the objectives, as outlined below.

C. Deep Optimistic Linear Support

One approach proposed by Mossalam et al. is the Deep Optimistic Linear Support Learning (Deep-OLS) algorithm [16]. This algorithm supports bi-objective optimization through scalarization. The approach first trains two Deep Neural Networks (DNNs) on the weight extrema, one optimized for each objective, and then uses the two DNNs to predict the utility of a solution. This is then used to create a Convex

Coverage Set (CCS), and applies Optimistic Linear Support (OLS) to select the weights for which a new policy could gain the most improvement in finding a new optimal policy. This results in multiple trained agents, each one optimal for a range of weights. These agents can then be swapped at runtime based on the weights required. The approach aims to minimize the amounts of networks trained to cover the weight space. This approach has been expanded upon for application on the placement problem. Due to the current limitations of the Deep-OLS algorithm, only two objectives are used. To keep the objectives as conflicting as possible, a focus was put on both a device and a network optimization objective. The network objectives were scalarized together and normalized to the range $[0, 1]$. This was subsequently also done for the device objectives. This approach reduces Eq. 5 to two objectives.

D. Conditioned Network

Our final approach uses a Conditioned Network (CN), removing the need of having multiple pre-trained agents completely, by introducing the weights into the observation space of the agent. As proposed by Abels et al. [17], the approach is based on an Universal Value Function Approximator (UVFA), where a network learns state and goal embeddings, using a distance-oriented metric to combine both. These goal embeddings are represented in MORL problems as the weight vectors. Training happens end-to-end, with the state and goal embedding as input and the multi-objective Q-values as output. In addition, a Diverse Experience Replay (DER) methodology is applied to the CN, which is a replay buffer which focuses on diversity. This is especially important, as the network should memorize the impact of the different weight vectors as well as the impact of the state-action pairs. Their approach is shown to have improved results compared to various other methodologies, including the multi-network approach.

VI. Evaluation

For evaluation, a use-case was crafted of 10 devices and 10 tasks, providing 10^{10} possible different placements. The networks and additions were built using RLLib [18]. Four objectives were evaluated: Energy, Worst Case Execution Time (WCET), Latency and Bandwidth. For the Deep-OLS approach, Energy and WCET were scalarized as device objective, and Latency and Bandwidth scalarized as network objective. Due to instability and slower convergence, the vector Q-values, proposed by Mossalam et al. were not used [16]. We expanded on the existing approaches of the Deep-OLS and Conditioned networks by building them using a Double Dueling DQN, which improves general stability and convergence. The hyperparameters used are found in Table I. The results were compared with an Non-dominated Sorting Genetic Algorithm II (NSGA-II) approach, as proposed in previous research [19]. This algorithm was configured with a population size of 100, running for 1000 iterations. Additionally, a comparison was made with a standard random search, iterating over 1000 possibilities before finishing.

TABLE I
HYPERPARAMETERS

γ	0.95
lr	0.00001
ϵ	150 000
batch size	32
buffer size	20 000
weight change interval	10 000 steps

VII. Results

All algorithms were generally able to find solutions that satisfied all constraints. Fig. 3 represents the average time required to find a single solution. We pooled the MORL algorithms, as they had similar networks and consequently similar inference time. The Cloud solution refers to placing all possible tasks on the cloud, showcasing the traditional approach. The log scale showcases that the proposed MORL approaches outperform the traditional NSGA-II algorithm by a factor 5. Note, however, that both NSGA-II and Random Search depend on the number of iterations to determine timing, whereas the proposed MORL algorithms have static timing and resource usage in nature. More interesting results are found on Fig. 4, which shows the average reward over 50 runs. The x-axis shows the weight for the network objective, where a 0 is the corner weight focusing on device objectives and 1 is the opposite corner weight focusing on network objectives. Note that the cloud solution does not satisfy the latency solution and is invalid, being purely shown as reference.
It is clear that the NSGA-II algorithm finds the optimal solution. This is at the tradeoff of consuming considerably more time and resources. The bi-objective Conditioned Network approach comes quite close to the NSGA-II algorithm, which showcases that a trained network is a valid approach in resource-constrained service placement. Interestingly, the neural network generally also finds better solutions in 50 timesteps than the random algorithm does in 1000. This is partially accredited to a light skew in the normalization, making network objectives slightly more valueable. In addition, if a corner weight of the Deep-OLS fails to converge, the subsequent search becomes infeasible. We notice that the conditioned network trained on four objectives succeeds at finding useful solutions, but is outperformed by nearly all other approaches. This is likely due to the large jump in complexity between solving for two and four objectives.

VIII. Discussion

The results showcase the brittleness of applying Deep-OLS in practical scenarios. The approach depends on finding the policies for the weight extrema first, but if these values are far apart, the algorithm stops working as expected. In addition, the approach suffers from search space complexity differences. In our scenario, it is considerably easier to optimize for network objectives, by putting all services on the same device, than

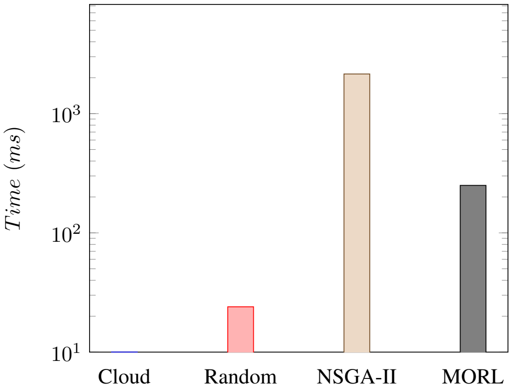

Fig. 3. Execution Time

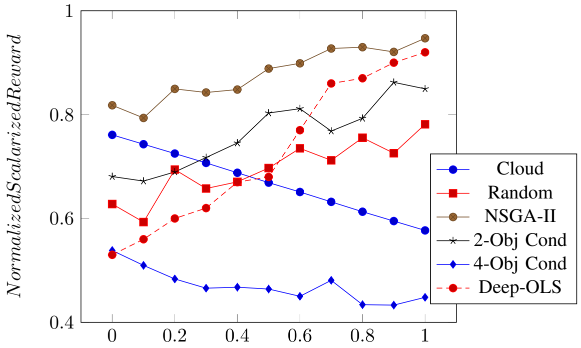

Fig. 4. Average Scalarized Reward

it is to optimize for device objectives. This mismatch makes it difficult to build an automated Deep-OLS search methodology, as the network objective policy converges considerably faster. We recommend to instead train individual policies with individual hyperparameters per weight and apply OLS on top.

IX. Future Work

Using the MORL techniques described, further objectives, such as privacy and security, and constraints, such as software requirements, can easily be added. The impact of these added objectives and constraints should be evaluated, and the scalability of the proposed techniques validated. The trained models could be further improved to reduce resource consumption and inference time, using network pruning, as proposed by Balemans et al. [20] In addition, our proposed technique can be extended to larger and varying computer networks or application chains. This can be tackled by using the strengths of Graph Neural Networks (GNNs), which allow neural networks to process graphs of any size [21].

References

[1] Cisco, "Cisco Anual Internet Report," Tech. Rep., 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges," pp. 3619–3647, 12 2017.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*. ACM Press, 2012, p. 13. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2342509.2342513

[4] F. A. Salaht, F. Desprez, and A. Lebre, "An Overview of Service Placement Problem in Fog and Edge Computing," 6 2020.

[5] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration Modeling and Learning Algorithms for Containers in Fog Computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 712–725, 9 2019.

[6] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach," *Future Generation Computer Systems*, vol. 110, pp. 1098–1115, 9 2020.

[7] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, "Intelligent Resource Allocation in Dynamic Fog Computing Environments," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*. IEEE, 11 2019, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/9064110/

[8] M. I. Naas, P. R. Parvedy, J. Boukhobza, and L. Lemarchand, "IFogStor: An IoT Data Placement Strategy for Fog Infrastructure," *Proceedings - 2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC 2017*, pp. 97–104, 2017.

[9] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, "Energy efficient task allocation and energy scheduling in green energy powered edge computing," *Future Generation Computer Systems*, vol. 95, pp. 89–99, 6 2019.

[10] C. Lee and Z. Ma, "The Generalized Quadratic Assignment Problem," no. 2, 3 2004. [Online]. Available: https://www.researchgate.net/publication/228575335

[11] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 4 2004. [Online]. Available: http://link.springer.com/10.1007/s00158-003-0368-6

[12] R. Kasimbeyli, Z. K. Ozturk, N. Kasimbeyli, G. D. Yalcin, and B. I. Erdem, "Comparison of Some Scalarization Methods in Multiobjective Optimization," *Bulletin of the Malaysian Mathematical Sciences Society*, vol. 42, no. 5, pp. 1875–1905, 9 2019. [Online]. Available: http://link.springer.com/10.1007/s40840-017-0579-4

[13] Z. Xia, Y. Chen, L. Wu, Y. C. Chou, Z. Zheng, H. Li, and B. Li, "A Multi-objective Reinforcement Learning Perspective on Internet Congestion Control," in *2021 IEEE/ACM 29th International Symposium on Quality of Service, IWQOS 2021*. Institute of Electrical and Electronics Engineers Inc., 6 2021.

[14] R. Rădulescu, P. Mannion, D. M. Roijers, and A. Nowé, *Multi-objective multi-agent decision making: a utility-based analysis and survey*, 2020, vol. 34, no. 1.

[15] S. K. Gottipati, Y. Pathak, R. Nuttall, Sahir, R. Chunduru, A. Touati, S. G. Subramanian, M. E. Taylor, and S. Chandar, "Maximum Reward Formulation In Reinforcement Learning," 10 2020. [Online]. Available: http://arxiv.org/abs/2010.03744

[16] H. Mossalam, Y. M. Assael, D. M. Roijers, and S. Whiteson, "Multi-Objective Deep Reinforcement Learning," 10 2016. [Online]. Available: http://arxiv.org/abs/1610.02707

[17] A. Abels, D. M. Roijers, T. Lenaerts, A. Now, and D. Steckelmacher, "Dynamic weights in multi-objective deep reinforcement learning," *arXiv*, 2018.

[18] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for Distributed Reinforcement Learning," 12 2017. [Online]. Available: http://arxiv.org/abs/1712.09381

[19] R. Eyckerman, S. Mercelis, J. Marquez-barja, and P. Hellinckx, "Evaluation of Objective Function Descriptions And Optimization Methodologies For Task Allocation In A Dynamic Fog Environment," 2020.

[20] D. Balemans, W. Casteels, S. Vanneste, J. de Hoog, S. Mercelis, and P. Hellinckx, "Resource efficient sensor fusion by knowledge-based network pruning," *Internet of Things (Netherlands)*, vol. 11, 9 2020.

[21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," 1 2019. [Online]. Available: http://arxiv.org/abs/1901.00596 http://dx.doi.org/10.1109/TNNLS.2020.2978386