QR-MO: Q-Routing for Multi-Objective Shortest-Path Computation in 5G-MEC Systems

Annisa Sarah
Dept. of Electrical Eng. and Computer Science
University of Stavanger
Stavanger, Norway
annisa.sarah@uis.no

Rosario G. Garroppo
Dept. of Information Eng.
University of Pisa
Pisa, Italy
rosario.garroppo@unipi.it

Gianfranco Nencioni
Dept. of Electrical Eng. and Computer Science
University of Stavanger
Stavanger, Norway
gianfranco.nencioni@uis.no

***Abstract*—Multi-access Edge Computing (MEC) is a promising technology that provides low-latency processing capabilities. To optimize the network performance in an MEC system, an efficient routing path between a user and its serving MEC host is essential. The network performance is characterized by multiple attributes, including packet-loss probability, latency, and jitter. A user service may require a particular combination of such attributes, complicating the shortest-path computation. This paper introduces Q-Routing for Multi-Objective shortest-path computation (QR-MO), which simultaneously optimizes multiple attributes. We compare the QR-MO's solutions with the optimal solutions provided by the Multi-objective Dijkstra Algorithm (MDA). The results show the favorable potential of QR-MO. After 100 episodes, QR-MO achieves 100% accuracy in networks with low to moderate average node degrees, regardless of the size, and more than 85% accuracy in networks with high average node degrees.**

***Index Terms*—multi-objective, shortest path, 5G, MEC.**

I. Introduction

THE integration of 5G-and-Beyond networks and Multi-access Edge Computing (MEC) is an example of integration between computing and communication. 5G-MEC systems aim to offer ultra-low latency high-bandwidth communication, real-time processing, and context awareness [1], [2]. An MEC system consists of several computing platforms positioned at the network edge, which are called MEC Hosts (MEHs). In an MEC system, user traffic should be allocated to the best path to the serving MEH to maintain a high-performance service. Most studies use the shortest path [3], [4]. However, shortest-path algorithms focus on one performance attribute and are not suited to provide heterogeneous services, as 5G is meant to. Network performance can be measured on the basis of different attributes, such as latency, jitter, and packet loss. A user may require different types of service, which may have stringent requirements on one of the attributes rather than another. Therefore, it is important to select a routing path that accounts for multiple cost attributes between a user and its serving MEH. This kind of problem is called Multi-Objective Shortest-Path (MOSP) problem.

A MOSP problem can be addressed in two different ways: (i) transforming the multiple objectives into a single combined objective [5]; (ii) generating the whole set of efficient paths as a reference for a decision maker [6]. The first approach computes the combined objective as a scalar function that integrates various objectives by assigning weights to each one based on their relative importance. The second approach is general and outputs a Pareto set, a set of optimal solutions that are non-dominated by each other and superior to the rest of the solutions. The second approach is preferable because the first approach requires the selection of the weight of each objective before solving the optimization problem. The first approach is not flexible because, in the case of a change of the weights, the problem needs to be solved from scratch.

The solution of MOSP problems can be calculated using optimal solution methods, such as ad-hoc mathematical programming method [5], [6], or approximation methods, such as Genetic Algorithm (GA) [7] or Reinforcement Learning (RL) [8]. The optimal solution methods are usually used for MOSP in specific network conditions and are not well-suited in a dynamic network scenario. Approximation methods are therefore preferred in today's networks that are highly dynamic. RL-based methods are preferred because GA usually needs a complicated representation and has scalability issues [7].

Regarding the RL-based solutions, Moffaert et al. [8] address general multi-objective optimization, not specifically solving the MOSP problem. In contrast, Yao et al. [9] address a specific MOSP problem related to route planning in smart cities, evaluating the efficacy of GA and RL, both of which are approximation methods. Rao et al. [10] use RL to solve dynamic MOSP problems, i.e., optimizing delay, packet loss and throughput, by transforming multi-objective to weighted single-objective. Although this approach performs well, it only returns one solution. This solution depends on the selected weights assigned to each attribute, rather than representing the Pareto front. To the best of our knowledge, there have been no efforts to evaluate the performance of approximating the Pareto front by using RL for MOSP problem in a 5G-MEC scenario and its effectiveness compared to optimal solutions.

This paper fills this gap by presenting an exploration of the use of RL to obtain an approximation of the Pareto front to solve an MOSP problem in 5G-MEC systems. We propose an RL-based approach, called *Q-Routing for Multi-Objective* shortest-path computation (*QR-MO*). Our initial study inves-

This work has been funded by the Norwegian Research Council through the 5G-MODaNeI project (no. 308909).

978-3-903176-75-1 ©2025 IFIP

tigates the accuracy of the solutions produced by QR-MO compared to the optimal solutions computed by a traditional deterministic algorithm, such as the Multi-Objective Dijkstra Algorithm (MDA). The study is carried out on multiple 5G-MEH networks under given conditions.

II. Related Works

The MOSP problem that seeks Pareto-optimal paths has been addressed through exact methods (e.g., labelling algorithms [11]) and heuristic strategies (e.g., evolutionary algorithms [12]). Raith et al. [11] compare exact algorithms (e.g., label-setting, dynamic programming) and empirically show their poor scalability in dense networks. Although these methods guarantee optimality, their computational overhead becomes prohibitive in environments such as 5G networks, where network conditions might change. This limitation motivates the need for lightweight, adaptive alternatives that can efficiently approximate the optimal Pareto set in real-time.

Recent studies demonstrate the use of RL to approximate the Pareto set in dynamic networks [13], [14]. These works provide foundational insights into handling multiple objectives in real-time scenarios, which is relevant in edge computing. However, machine learning methods for multi-objective optimization, including RL, have advanced broadly, but no effort has been made for MOSP problems in 5G-MEC systems.

There have been several efforts instead to use RL to compute paths by optimizing a single objective. One specific simplified RL approach is called Q-learning. Q-learning has been adapted for routing since Boyan and Littman's Q-routing framework [15] optimizes single-objective paths. However, there is a lack of exploration in using Q-routing for multi-objective path optimization. While some works have applied classical Q-learning to routing problems, they often fail to address the complexities of multi-objective scenarios or leverage improved versions tailored for such tasks [16]. This gap highlights the need for more advanced methods capable of handling multiple objectives efficiently, particularly in 5G-MEC environment given the unique constraints of the service deployed in 5G-MEC systems [17].

In 5G-MEC systems, delivering Ultra-Reliable Low-Latency Communication (URLLC) requires routing solutions that balance multiple objectives, such as latency, jitter, and packet loss, while adapting to network conditions. While single-objective approaches minimize latency [18], they ignore Pareto optimality. Multi-Dijkstra algorithms [6] compute the exact Pareto set but are computationally prohibitive in dynamic 5G networks. RL-based methods such as Q-Routing [15] adapt to network changes but focus on single objectives, leaving a gap in joint optimization of path in 5G-MEC networks.

Our work is the first effort to solve a MOSP problem in 5G-MEC systems using RL and has the following contributions:

- **Extension of Q-Routing to Multi-Objective:** while Q-Routing [15] was designed for single-objective optimization, we adapt it to MOSP through heuristic action selection. This is the first adaptation of Q-Routing to approximate a Pareto set.

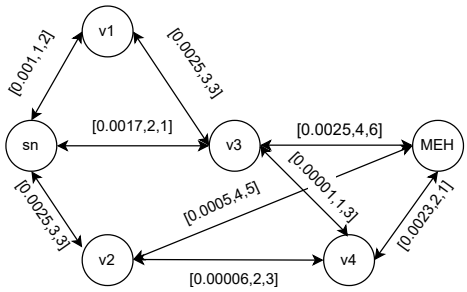

Fig. 1. Illustration of a MOSP problem

- **Preliminary Validation of the Accuracy:** QR-MO is evaluated by computing the proximity to optimal solutions, i.e. the Pareto set computed by MDA.
- **Context-aware Path Computation in 5G-MEC system:** The evaluation is performed on various 5G-MEC networks considering heterogeneous service requirements for latency, jitter, and packet loss.

III. Problem Definition and MDP Representation

A MOSP problem can be illustrated as in Fig. 1. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $v \in \mathcal{V}$ and edges $e \in \mathcal{E}$, each edge has J performance attributes, which are also called cost attributes, $\mathbf{c}_e = \{c_{e1}, c_{e2}, \ldots, c_{eJ}\}$. In this and the following sections, we do not define any specific cost attribute to keep the problem formulation and solution general. Specific cost attributes will be presented at the beginning of Section V. Solving the MOSP problem means finding a path that optimizes the different cost attributes, which are often conflicting. We have to find a set of strictly non-dominated sets, i.e., the Pareto set.

Defined $P_{(sn,m)}$ and $Q_{(sn,m)}$ as two paths from the source node sn to the MEH m and defined $c_j(P)$ and $c_j(Q)$ as the j-th attribute of the cost vector for paths P and Q respectively, the path $P_{(sn,m)}$ dominates the path $Q_{(sn,m)}$, denoted as $P_{sn,m} \prec_D Q_{sn,m}$, if the following condition is valid:

$$\begin{aligned} P_{sn,m} \prec_D Q_{sn,m} \iff \\ (c_j(P_{sn,m}) \leq c_j(Q_{sn,m}) \quad \forall j \in [1 \,..\, J]) \\ \wedge (\exists k \in [1 \,..\, J] : c_k(P_{sn,m}) < c_k(Q_{sn,m})) \end{aligned} \tag{1}$$

This means that each of the cost attributes of path $P_{(sn,m)}$ is less than or equal to those of path $Q_{(sn,m)}$. Furthermore, there exists at least one cost attribute path $P_{(sn,m)}$ that is strictly less than the one of path $Q_{(sn,m)}$.

Defined $P_{(sn,m)}$ as a path from the source node sn to node m.

$$\min \mathbf{F}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_J(\mathbf{x}) \end{bmatrix}, \tag{2}$$

The objective of the MOSP problem is to minimize all cost attributes associated with the selected path, as expressed in

Eq. 2, where the cost function for each attribute j is defined as follows.

$$f_j(\mathbf{x}) = \sum_{e \in \mathcal{E}} c_{ej} \cdot x_e, \quad j = 1, 2, \ldots, J. \tag{3}$$

In Eq. 3, the function $f_j(\mathbf{x})$ calculates the total cost for the j-th cost attribute across the selected edges in the path. Specifically, for each edge $e \in \mathcal{E}$, the cost c_{ej} associated with the j-th attribute is weighted by the decision variable x_e, which indicates whether the edge e is included in the path ($x_e = 1$) or not ($x_e = 0$).

The optimization is subject to the following constraints:

$$x_e \in \{0, 1\}, \quad \forall e \in \mathcal{E}. \tag{4}$$

$$\sum_{e \in \text{Adj}(v)} x_e = \begin{cases} 1, & \text{if } v = sn, \\ 1, & \text{if } v = m, \\ 2, & \text{if } v \in \mathcal{V} \notin \{sn, m\}, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Eq. 4 ensures that each edge $e \in \mathcal{E}$ is either selected or not, represented by the binary decision variable x_e.

Eq. 5 shows the flow conservation constraint that ensures that the selected edges form a valid path. At the source node sn, exactly one edge should be selected for the path. Similarly, at the destination node, exactly one edge should be selected for the path. For any intermediate node, exactly two edges must be selected. Lastly, nodes that are not part of the path have no selected edges.

IV. Proposed Solution

RL traditionally optimizes a single scalar reward, yet many real-world problems involve multiple conflicting objectives [19]. Multi-objective RL can be categorized as: (i) utility-based vs Pareto-based, and (ii) single-policy vs multi-policy [19].

Utility-based approaches focus on optimizing a single scalarized reward function, which combines multiple objectives into a weighted sum or other forms of aggregation. Although this simplifies the optimization problem, it requires predefined weights or utility functions, which are often difficult to determine [10], [20]. On the other hand, Pareto-based approaches aim to identify a set of Pareto-optimal solutions, each representing a trade-off where improving one objective would degrade another. This provides decision-makers with more flexibility in choosing the most appropriate solution based on specific preferences or priorities [21]–[23].

Single-policy methods aim to learn a single optimal solution, typically tailored to a fixed utility function. However, these methods do not accommodate uncertain preferences, which limits their applicability in real-world scenarios [19]. In contrast, multi-policy approaches generate a diverse set of solutions, offering greater adaptability to varying or evolving preferences, which is crucial in environments with competing objectives or changing constraints [23].

By leveraging Pareto-based and multi-policy approaches, multi-objective RL can better address the complexities of real-world decision making, providing not only a broader set of solutions but also more robust adaptability to diverse needs. The proposed QR-MO algorithm builds on these principles by approximating the Pareto front and adopting a multi-policy approach. Using RL and heuristic action selection, QR-MO identifies a diverse set of Pareto-optimal solutions, each representing a trade-off between conflicting objectives. This multi-policy capability ensures that QR-MO adapts effectively to evolving preferences in real-time environments, such as 5G-MEC systems, where conditions and requirements are constantly changing.

A. Q-learning

Q-learning is a model-free algorithm that learns the state-action value (or Q value). The Q-value represents the expected total value for a particular action a in a given state s. The Q-value is generally updated according to the update rule in Eq. 6. To update the Q-values, we have to sum three factors: the current values $(1 - \alpha) \cdot Q(s, a)$, the reward r of taking action a from state s, and the maximum reward that can be obtained from next state s', $\max_a Q(s', a')$. The Q-learning algorithm stores the state-action values in a Q-table, which will be updated until they converge or certain criteria are met.

$$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_a Q(s', a')) \tag{6}$$

$$Q(s, a) \leftarrow (1-\alpha) \cdot Q(s, a) + \alpha \cdot (\mathbf{c_{s,a}} + \min_{a' \in \text{neighbors of } s'} Q(s', a')) \tag{7}$$

Boyan and Littman [15] proposed a modified Q-learning for a routing problem, namely Q-routing. The Q-routing does not need the discount factor as in the generic Q-learning technique ($\gamma = 1$). The Q-routing aims to minimize the future cost (i.e., minimize the total latency) instead of maximizing the future reward. For the Q-routing, the Q-update equation will be slightly changed as in Eq. 7. The state s is the current node v, where $|S| = |\mathcal{V}|$ and the action a is the edge e from the current node minus the incoming edge. The Q-values of state s, taking action a that leads to the next state s', are updated by the sum of the current value, the cost attributes $\mathbf{c_{s,a}}$ of the action a, and the minimum cost attributes of the neighbors of the next state s' to reach the end node.

B. QR-MO

The problem with the classical Q-routing technique is that it can only consider one cost. Algorithm 1 shows the modified Q-routing algorithm that addresses the multi-objective problem, namely *Q-Routing for Multi-Objective* problem (*QR-MO*). The Q-routing implementation in our work is a modification of the code from [24]. In contrast, the Q-table in the proposed QR-MO stores multiple values for each $Q(s, a)$, corresponding to the various cost attributes considered. To choose the best action a in each state s, the QR-MO simultaneously considers different criteria by using a heuristic approach. Consequently, QR-MO can generate J solutions, i.e., each solution is the best solution for a specific attribute.

Algorithm 1 presents the detailed steps for the QR-MO. We first initialize a network graph G, set a start node n^S

Algorithm 1 QR-MO for Path Selection

Initialize: Load the network graph $\mathcal{G}$; A start node n^S and end node n^E; Number of episodes N; Cost matrix R; Uniform Q; Empty memory $\mathcal{B}$; Learning rate α; ϵ for the epsilon-greedy action selection.
for i **to** N **do**
  Initialize $t = 0$
  Initialize the current state to the start node $s_t = n^S$
  while $s_t \neq n^E$ **do**
    Check next possible nodes from s_t
    Choose a_t by using an ϵ-greedy policy as below:
    $a_t = \begin{cases} \text{random } a, & \text{if } p = \epsilon \\ \text{DominanceSelection}(a_{t-1}, s_t, Q_t), & \text{if } p = 1 - \epsilon \end{cases}$
    Perform the chosen action a_t, transition to s_{t+1}
    $Q(s_t, a_t) \leftarrow$ UpdateQ(R, Q_t, s_t, a_t, α)
    Update the current state $s_t = s_{t+1}$
  Get the route $l_i \leftarrow \{n^S, ..., n^E\}$ and Q-values Q_i
  Check and store the best cost and path $\mathbf{B} \leftarrow$ UpdateBestPath(l_i, G,$\mathbf{B}$,Q_i,i)
Return: (1) Q (2) $\mathbf{B}$

and an end node n^E, a cost matrix R, uniform Q-values for all pairs of state $s \in S$ and action $a \in A$, and an empty variable B to store the record of best Q-values, best paths and best cost of each cost attribute j while learning throughout episodes. We also set the RL hyper-parameters: learning rate α and epsilon greedy parameter ϵ. Then, for each episode, the QR-MO agent learns the path from starting node n^S to reach end node n^E by selecting a proper action a on the current state s. We employ a ϵ-greedy policy to select an action, meaning that the QR-MO agent will take a random action with probability ϵ and a greedy action (i.e., take the best action) with probability 1-ϵ. The ϵ-greedy policy is useful to balance the exploration and exploitation of the learning to seek an optimal policy. The best action for the QR-MO can be selected by using a heuristic algorithm called DominanceSelection. Algorithm DominanceSelection has been adapted from the paper [6] and used to evaluate the domination of the cost attributes of neighbouring edges.

Algorithm 2 DominanceSelection

Input: Previous selected action a_{t-1}; Current state s_t; Learned policy $\mathbf{Q}$.
Initialize: Defined $\mathcal{A}_{keys}$ as the set of the possible actions (edges to neighboring nodes) from state s_t excluding the previous selected action a_{t-1}, i.e., incoming edge; An empty dictionary D to store dominance scores for each action $D(a) = 0 \quad \forall a \in A_{keys}$
for each $(a_x, a_y) \in C_{pairs}$ **do**
  for each cost index j **do**
    if $Q^j(s_t, a_x) \leq Q^j(s_t, a_y)$ **then**
      Increment $D(a_x)$ by 1
    if $Q^j(s_t, a_x) > Q^j(s_t, a_y)$ **then**
      Increment $D(a_y)$ by 1
Select the best action:
$a_{dom} \leftarrow \arg\max_{a \in A_{keys}} D(a)$
Return: a_{dom}

Algorithm DominanceSelection needs three inputs: previous selected action a_{t-1}, current state s_t, and Q-values at timestep t, Q_t. The previously selected action a_{t-1} is the action that made the agent visit the current state s_t, i.e., the incoming edge. We initialize a dictionary D to count the dominance scores for each action. Then, all possible actions from current state $a \in \mathcal{A}_{keys}$ are categorized as pairs $(a_x, a_y) \in C_{pairs}$ where $x \in \mathcal{A}_{keys}, y \in \mathcal{A}_{keys}, x \neq y$. All possible actions $\mathcal{A}_{keys}$ are all edges connecting to the current node s_t, except the incoming edge, which is the previously selected action a_{t-1}. For each pair (a_x, a_y), we investigate the dominance based on each cost attribute j and count the dominance scores of all actions. The best action a_{dom} is the one with the highest scores among all possible actions $\mathcal{A}_{keys}$ and is returned to Algorithm 1.

Algorithm 3 UpdateBestPath

Input: Network graph $\mathcal{G}$; Q-values matrix for episode i Q_i; Start node n^S and end node n^E; Route of current episode i, l_i from n^S to reach end n^E, Stored memory $\mathbf{B}$=[$\{l_1^B, \mathbf{c}^{l_1^B}, Q_1^B\}, \{l_2^B, \mathbf{c}^{l_2^B}, Q_2^B\}, \{l_3^B, \mathbf{c}^{l_3^B}, Q_3^B\}$]
Initialize: Costs of route l_i, $\mathbf{c}^{l_i} = \{c_1^{l_i}, c_2^{l_i}, c_3^{l_i}\}$
for each j in c_j **do**
  if $c_j^{l_i} < c^{l_j^B}$ **then**
    Update best cost j on memory $c^{l_j^B} \leftarrow c_j^{l_i}$
    Update best route for cost j on memory $\mathbf{c}^{l_j^B} \leftarrow l_i$
    Store $Q_j^B \leftarrow Q_i$
return updated $\mathcal{B}$

In Algorithm 1, the Q-value $Q(s_t, a_t)$ is then updated using Eq. 7. After reaching the end node n^E, the route l_i and Q-values Q_i are stored. Lastly, we have to check the best cost, route, and Q-values B by using the Algorithm UpdateBestPath. Algorithm UpdateBestPath evaluates the learned route, cost, and Q-values on episode i by comparing it with the tuple $\{l_j^B, \mathbf{c}^{l_j^B}, Q_j^B\}$, where l_j^B is the current best route of attribute j, $\mathbf{c}^{l_j^B}$ is the tuple of all cost attributes concerning the route l_j^B, and Q_j^B is the Q-values or policy to generate the route l_j^B. The stored memory is then compared. If the learned cost of attribute j on episode i is better than the best cost j in the memory B, then the memory is updated. This algorithm ensures that the QR-MO returns multiple solutions, considering all cost attributes simultaneously, compared to classic Q-routing, which only returns one solution.

C. Complexity Analysis

QR-MO operates on a graph with $|\mathcal{V}|$ nodes, $|\mathcal{E}|$ edges, and J cost attributes. It iterates over N episodes, refining paths via a While loop that explores the graph and selects actions at each node. Algorithm DominanceSelection compares action pairs, requiring $O(d^2 \cdot J)$ operations per node with degree d. Since traversal involves up to $|\mathcal{V}|$ nodes, the per-episode complexity is $O(|\mathcal{V}| \cdot J \cdot d^2)$, where $d \leq |\mathcal{E}|/|\mathcal{V}|$. Algorithm UpdateBestPath evaluates paths with $O(J)$ operations per path, which is negligible compared to selection. Thus, the total complexity of QR-MO is $O(N \cdot |\mathcal{V}| \cdot J \cdot d^2)$. Substituting $d^2 \sim (|\mathcal{E}|/|\mathcal{V}|)^2$, it simplifies to $O(N \cdot J \cdot |\mathcal{E}|^2/|\mathcal{V}|)$. For sparse

graphs ($|\mathcal{E}| \sim |\mathcal{V}|$), this reduces to $O(N \cdot |\mathcal{V}| \cdot J)$, while for dense graphs ($|\mathcal{E}| \sim |\mathcal{V}|^2$), it scales as $O(N \cdot |\mathcal{V}|^3 \cdot J)$.

Comparing QR-MO to existing Pareto optimal algorithms, such as MDA, highlights key efficiency trade-offs. As presented by Casas et al. [6], MDA maintains multiple non-dominated labels per node. Let L be the average number of labels per node and $L_{\max}$ be the maximum number of labels at any node, then MDA has a complexity equal to $O\left(|\mathcal{V}| \cdot J\left(L \cdot \log|\mathcal{E}| + L_{\max}^2 \cdot |\mathcal{E}|\right)\right)$.

Heap operations introduce a logarithmic term, while dominance checks scale quadratically with $L_{\max}^2$ when $J \geq 3$, making MDA expensive for high-dimensional cases (i.e., scenarios where the number of cost attributes J is high). As J increases, the number of non-dominated solutions grows, making dominance checks more computationally expensive. The term $L_{\max}^2 \cdot |\mathcal{E}|$ dominates for high J ($J \geq 3$). For high J, the number of stored labels per node increases due to the growth in non-dominated solutions. The dominance check in MDA requires comparing each label against others, leading to a worst-case complexity of $O(L_{\max}^2)$ per edge. Since $L_{\max}$ increases with J, the term $L_{\max}^2 \cdot |\mathcal{E}|$ dominates, making MDA less efficient in high-dimensional settings.

In practical applications, QR-MO is preferable for sparse graphs or high J, avoiding quadratic label growth that hampers MDA in multi-objective settings. MDA remains effective in low-dimensional, highly connected graphs, where efficient queue operations mitigate dominance check overhead.

V. Experimental Results

We have developed our simulator that takes the network graphs from the dataset [25], which has three synthetic graphs and one real network scenario for a 5G-MEC system in Milan City Centre (MCC). The MCC graph has been interpolated from OpenCellID. There are four network topologies, i.e., 25N50E, 100N150E, 30N35E, and 50N50E. 25N50E means that the network consists of 25 nodes and 50 edges. These network topologies have been selected because they represent a 5G-MEC system with a variety of network characteristics. 25N50E and 100N150E have a high average node degree, with 3 or 4 edges per node, respectively. MCC and 50N50E have a low average node degree of 2.3 and 2, respectively.

The dataset provides only the network structural information and assumes the same cost for all edges. We have modified the cost values and consider *three cost attributes* that are randomly generated using a uniform distribution: (1) *packet-loss probability* with a Probability Density Function (PDF) of 1/3 $U(0.0005, 0.1)$+ 2/3 $U(0, 0.0005)$), (2) *latency* 1/3 $U(5, 10)$ + 2/3 $U(1, 5)$) ms, and (3) *jitter* 1/3 $U(3, 5)$ + 2/3 $U(1, 3)$). The packet loss, latency, and jitter are taken from measurements in a 5G Campus Networks [26]. Rischke et al. [26] indicate that the packet loss probability of a single transmission in a testbed of non-standalone 5G, with a packet size of 128 to 256 bytes, ranges between 0 and 0.15. The latency of the same type of network varies between 1 ms and 10 ms [26]. For jitter, the values are derived from [26] and [27], which show that it varies from 1 ms to half of the maximum latency.

A. Performance Metrics

Since QR-MO is the first RL-based approach to compute the Pareto set for MOSP problems in 5G-MEC systems, we compare the approximated Pareto set computed by QR-MO with the optimal Pareto set computed with MDA. To compare the two Pareto sets and evaluate the effectiveness of QR-MO, we have used two performance metrics.

An evaluation metric called *Distance to Pareto Set (DPS)* is computed to determine the proximity of the QR-MO solution to the Pareto set. The DPS consists of the Euclidean distance between the QR-MO solutions and the Pareto set. The i-th solution of the Pareto set is denoted as $\mathbf{\Psi_i} = \{\Psi_{ij} \quad \forall j \in [1 \,..\, J]\}$, where Ψ_{ij} is the value of the j-th cost attribute. There is no predetermined number of solutions in the Pareto set, i.e., the number of i-es is unknown. Instead, QR-MO always returns K solutions, and the k-th solution is denoted as $\mathbf{\Omega_k} = \{\Omega_{kj} \quad \forall j \in [1 \,..\, J]\}$. Therefore, in QR-MO, the number of solutions is predetermined and is equal to the number of cost attributes $K = J = 3$ because each of the QR-MO solutions optimizes one of the attributes. As previously mentioned, the cost attributes considered in this study are packet loss probability, latency, and jitter.

The DPS can be calculated as follows. First, given the different scales of the cost attributes, the value of each attribute is normalized to the range $[0, 1]$. For each attribute $j \in [1 \,..\, J]$, the normalization factor f_j is calculated as $f_j = \max_{i,k}\{\Psi_{ij}, \Omega_{kj}\}$. The QR-MO solutions and each solution of the Pareto set are normalized by using the corresponding normalization factor f_j. The normalized solutions are obtained as $\mathbf{\Omega_k^{norm}} = \{\frac{\Omega_{k1}}{f_1}, \cdots, \frac{\Omega_{kj}}{f_J}\}$ and $\mathbf{\Psi_i^{norm}} = \{\frac{\Psi_{i1}}{f_1}, \cdots, \frac{\Psi_{ij}}{f_J}\}$. The distance between two solutions d_{ki} is calculated as $d_{ki} = \sqrt{\sum_{j=1}^{J}\left(\Omega_{kj}^{norm} - \Psi_{ij}^{norm}\right)^2}$. DPS is the minimum distance between the two sets of solutions $DPS = \min_{k,i} d_{ki}$. The lower DPS, the closer QR-MO solutions are to Pareto set.

The other performance metric is the *correctness*, which is used to assess whether one of the QR-MO solutions is part of the Pareto set; if it is true, then the correctness is equal to 1. The *average correctness* is the mean of the correctness of QR-MO solutions throughout all simulation runs. Since QR-MO returns K solutions, i.e., one for each cost attribute, it is important to evaluate how many solutions of the Pareto set QR-MO can find. The *average number of correct solutions* shows the mean number of QR-MO solutions that are part of the Pareto set.

B. Simulation Setting and Result Discussion

We aim to compare our QR-MO solution to its optimality. Therefore, we use MDA as a baseline algorithm to evaluate our proposed solution, since it returns optimal solutions. MDA is a label-setting algorithm introduced in the paper [6] to address the MOSP problem. MDA generates all solutions in the Pareto set, identified through the list of non-dominated labels of the nodes. The concept of non-domination is explained in Section III. MDA operates under the assumption that the cost attributes of each edge are summable. While this is true for

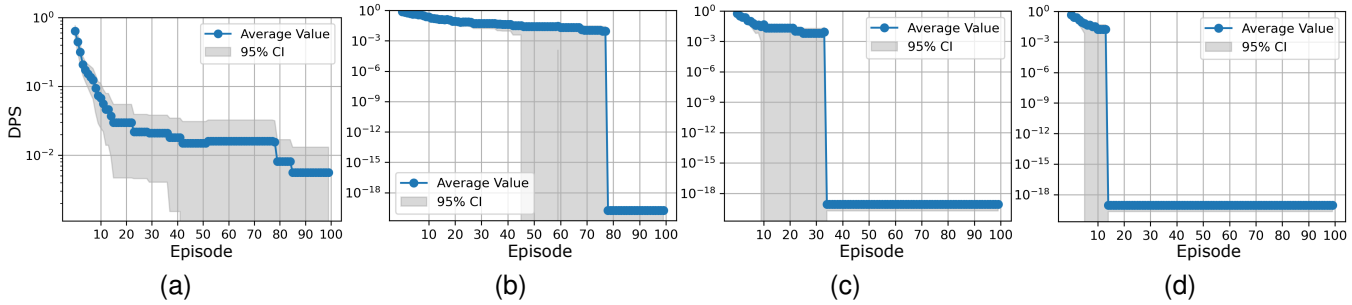

Fig. 2. Average DPS of QR-MO of network (a) 25N50E, (b) 100N150E, (c) MCC (30N35E), and (d) 50N50E

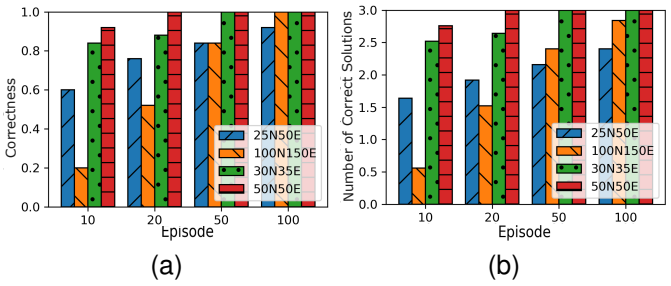

Fig. 3. Comparison of (a) average correctness of QR-MO and (b) the average number of correct solutions of QR-MO for different topologies

latency and jitter, it does not apply to packet loss probability. To address this limitation, the packet loss probability must be converted to logarithmic form to become summable.

The QR-MO uses an ϵ-greedy action selection with $\epsilon = 0.1$ and the learning rate $\alpha = 0.7$. The hyperparameters of QR-MO are decided after the empirical studies, and the selected values give the best overall performance. A QR-MO agent starts exploring the network from the start node to the end node. We set the maximum episode number N to 100, and one episode refers to a period when a QR-MO agent has a single exploration from a starting node and reaches an end node. We generate 25 different instances: 5 pairs of start and end nodes, and for each pair, we conduct 5 simulation runs. On each run, we have a randomization of ϵ greedy action selection; thus, we show the average and the 95% confidence interval. The experiments are performed on a laptop equipped with 8 virtual CPUs, a 2.8 GHz processor, 32 GB of RAM, and Python 3.9. Fig. 2 illustrates the DPS across the various network topologies over $N = 100$ episodes. The QR-MO performs exceptionally well for both the MCC and 50N50E. The DPS decreased to nearly 0, with around 30 episodes on the MCC network and around 10 episodes on the 50N50E network. MCC and 50N50E are small networks with low node degrees of 2.3 and 2.0 edges per node, resembling a tree structure. In such tree-like topologies, the QR-MO agent can quickly learn the optimal path due to fewer choices at each node.

On the other hand, for 25N50E and 100N150E, the DPS reaches 10^{-2} and near 0, respectively, after 100 episodes. This means that QR-MO achieves a near-optimal solution. These networks have high average node degrees of 4.0 and 3.0, resembling mesh topologies. The QR-MO agent requires more episodes in such high-degree networks to identify the paths. Nevertheless, the decrease in DPS from 10^{-1} to near zero within 80 episodes for both networks highlights a good potential of RL algorithms also for mesh networks. For 100N150E with a moderate average node degree, QR-MO can return a near-optimal solution with DPS $= 10^{-18}$.

Fig 3(a) illustrates the average correctness at episodes 10, 20, 50, and 100. Fig 3(b) depicts the average number of QR-MO solutions within the Pareto set. The higher episode improves the correctness and average number of correct solutions. In 50N50E, QR-MO has the best performance, reaching 100% correctness in 20 episodes, followed by the MCC network, which reaches 100% correctness in 50 episodes. QR-MO performs better in the 50N50E network than in the slightly denser MCC network, with the lowest average node degree of 2 edges per node. Nonetheless, after 50 episodes, all QR-MO solutions from both networks are correct.

Regarding the correctness metric, QR-MO produces near-optimal correctness for the 25N50E and 100N150E networks in 100 episodes. Both networks show an increasing trend in correctness with more episodes. In the 25N50E network, QR-MO initially achieves an average correctness higher than the one in the 100N150E network, around 60% instead of 20% correctness at episode 10. Anyway, when the number of episodes increases, QR-MO performs better in the 100N150E network (100% correctness and an average number of correct solutions of around 2.8 at episode 100) than in the 25N50E network (88% correctness at episode 100). The 40% gap difference in correctness at 10 episodes between 25N50E and 100N150E is due to the total number of nodes difference, with the 100N150E having four times more nodes. However, at episode 100, the correctness of QR-MO in 100N150E is 10% better than that in 25N50E. This indicates that the number of nodes significantly affects the average correctness, particularly with fewer episodes. With more episodes (50,100), the performance difference is more impacted by the average node degree, and the lower average node degree network can outperform the higher one.

The results presented in Fig. 2 and Fig. 3 indicate that the RL algorithm has promising potential for solving the MOSP problem and achieving near-optimal solutions in both small and large networks. By episode 100, the QR-MO algorithm had found near-optimal solutions across different networks. Anyway, QR-MO's processing time is in the order of thou-

sands of milliseconds, whereas MDA generates solutions in hundreds of milliseconds. However, this refers to a static network condition, i.e., the MOSP problem is solved for a given combination of cost values, nodes, and edges.

VI. Conclusion and Future Works

This paper explores the performance of an RL-based approach to solve the MOSP problem in 5G-MEC systems. QR-MO is the proposed RL-based approach, which modifies Q-routing to accommodate multiple objectives and uses a heuristic procedure for selecting the action and storing the solutions. QR-MO solutions has been compared with the optimal solutions provided by MDA. We have introduced two performance metrics to evaluate QR-MO performance: DPS and correctness. Four networks with different numbers of nodes and different average node degrees have been considered. In the case of tree-like networks with a small average node degree, QR-MO performs best in terms of DPS and correctness: fewer episodes can return correct solutions. QR-MO also performs well in the case of a mesh network but can return correct solutions after a higher number of episodes. However, our work still has some limitations and challenges. Although QR-MO has promising results, the processing time to reach the correct solution is hundreds to thousands of milliseconds. Meanwhile, MDA can generate solutions in tens to hundreds of milliseconds. Anyway, our evaluation is with static network conditions, further works need to be done to evaluate the profitability of QR-MO in dynamic networks. In dynamic network conditions, MDA must recompute the solutions from scratch at every network change. QR-MO can instead exploit the previous training to compute the solutions in the new network conditions. In this case, after convergence, QR-MO computes solutions for each episode, even when network conditions differ from those used during training and previous episodes. The QR-MO can implement its learned policy directly, generating solutions in about 5 ms on average. This suggests that QR-MO has the potential to adapt to dynamic changes in the network and adjust its policy in near real-time. In contrast, the MDA, which depends on static network assumptions, must generate solutions from scratch, requiring approximately 50 ms.

References

[1] ETSI, "GR MEC 031: 5G MEC Integration V.2.1.1," Dec. 2020.

[2] A. Sarah, G. Nencioni, and M. M. I. Khan, "Resource allocation in multi-access edge computing for 5g-and-beyond networks," *Computer Networks*, vol. 227, 2023.

[3] A. Buzachis, A. Celesti, A. Galletta, J. Wan, and M. Fazio, "Evaluating an application aware distributed dijkstra shortest path algorithm in hybrid cloud/edge environments," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 2, 2021.

[4] M. R. Anwar, S. Wang, M. F. Akram, S. Raza, and S. Mahmood, "5g-enabled mec: A distributed traffic steering for seamless service migration of internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 1, 2022.

[5] G. H. Shirdel and S. Ramezani-Tarkhorani, "A dea-based approach for finding a favorable multi-objective shortest path," *Croatian Operational Research Review*, vol. 9, no. 2, 2018.

[6] P. M. de las Casas, A. Sedeno-Noda, and R. Borndörfer, "An improved multiobjective shortest path algorithm," *Computers & Operations Research*, vol. 135, 2021.

[7] A. Uthayasuriyan, H. Chandran, U. Kavvin, S. H. Mahitha, and G. Jeyakumar, "A comparative study on genetic algorithm and reinforcement learning to solve the traveling salesman problem," *Research Reports on Computer Science*, 2023.

[8] K. Van Moffaert and A. Nowé, "Multi-objective reinforcement learning using sets of pareto dominating policies," *The Journal of Machine Learning Research*, vol. 15, no. 1, 2014.

[9] Y. Yao, Z. Peng, B. Xiao, and J. Guan, "An efficient learning-based approach to multi-objective route planning in a smart city," in *2017 IEEE Int. Conference on Communications (ICC)*. IEEE, 2017.

[10] Z. Rao, Y. Xu, Y. Yao, and W. Meng, "Dar-drl: A dynamic adaptive routing method based on deep reinforcement learning," *Computer Communications*, vol. 228, 2024.

[11] A. Raith and M. Ehrgott, "A comparison of solution strategies for biobjective shortest path problems," *Computers & Operations Research*, vol. 36, no. 4, 2009.

[12] F.-S. Chang, J.-S. Wu, C.-N. Lee, and H.-C. Shen, "Greedy-search-based multi-objective genetic algorithm for emergency logistics scheduling," *Expert Systems with Applications*, vol. 41, no. 6, 2014.

[13] C. Zhang, X. Song, Z. Liu, B. Ma, Z. Lv, Y. Su, G. Li, and Z. Zhu, "Real-time and multi-objective optimization of rate-of-penetration using machine learning methods," *Geoenergy Science and Engineering*, vol. 223, 2023.

[14] M. A. Khamis and W. Gomaa, "Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework," *Engineering Applications of Artificial Intelligence*, vol. 29, 2014.

[15] J. Boyan and M. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in Neural Information Processing Systems*, J. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan-Kaufmann, 1993.

[16] J. Liu, Q. Wang, C. He, K. Jaffrès-Runser, Y. Xu, Z. Li, and Y. Xu, "Qmr: Q-learning based multi-objective optimization routing protocol for flying ad hoc networks," *Computer Communications*, vol. 150, 2020.

[17] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, 2019.

[18] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, 2017.

[19] C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz *et al.*, "A practical guide to multi-objective reinforcement learning and planning," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 1, 2022.

[20] R. Rădulescu, P. Mannion, Y. Zhang, D. M. Roijers, and A. Nowé, "A utility-based analysis of equilibria in multi-objective normal-form games," *The Knowledge Engineering Review*, vol. 35, 2020.

[21] I. Mehta, S. Taghipour, and S. Saeedi, "Pareto frontier approximation network (pa-net) to solve bi-objective tsp," in *2022 IEEE 18th Int. Conference on Automation Science and Engineering (CASE)*. IEEE, 2022.

[22] J. Perera, S.-H. Liu, M. Mernik, M. Črepinšek, and M. Ravber, "A graph pointer network-based multi-objective deep reinforcement learning algorithm for solving the traveling salesman problem," *Mathematics*, vol. 11, no. 2, 2023.

[23] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, 2020.

[24] S. Yuan, "Reinforcement-learning-in-path-finding," 2023. [Online]. Available: https://github.com/shiluyuan/Reinforcement-Learning-in-Path-Finding

[25] B. Xiang, J. Elias, F. Martignon, and E. Di Nitto, "A dataset for mobile edge computing network topologies," *Data in Brief*, vol. 39, 2021.

[26] J. Rischke, P. Sossalla, S. Itting, F. H. Fitzek, and M. Reisslein, "5g campus networks: A first measurement study," *IEEE Access*, vol. 9, 2021.

[27] F. Ronteix-Jacquet, "Reducing latency and jitter in 5G radio access networks," Theses, Ecole nationale supérieure Mines-Télécom Atlantique, Dec. 2022.