Learning to Mine: Integrating Channel Quality Patterns for Enhanced AI-Assisted Scheduling Decisions in 6G Multimedia Networks

Ioan-Sorin Comșa[1], Per Bergamin[1], Gabriel-Miro Muntean[2], Purav Shah[3], and Ramona Trestian[3]
[1]Institute for Research in Open-, Distance- and eLearning,
Swiss Distance University of Applied Sciences, CH-3900, Brig, Switzerland,
[2]School of Electronic Engineering, Dublin City University, D09 V209, Dublin, Ireland,
[3]London Digital Twin Research Centre, Middlesex University London, NW4 4BT, Hendon, London, U.K.
E-mails: {ioan-sorin.comsa, per.bergamin}@ffhs.ch, gabriel.muntean@dcu.ie, {p.shah, r.trestian}@mdx.ac.uk

***Abstract*—This paper addresses the growing complexity of 6G Radio Resource Management (RRM) due to the integration of new multimedia services and sophisticated network functionalities. To handle complex RRM control issues, machine learning is recognized as a valuable tool for improving decision-making in scheduling. However, the effectiveness of intelligent scheduling methods can be limited without efficiently managing the large volumes of data produced by network measurements, such as Channel Quality Indicator (CQI) reports. To tackle this, we propose a low-complexity framework that compresses extensive CQI data into a compact, usable form for machine learning applications. The framework adopts a metaheuristic approach with simulated annealing to perform clustering and identify global CQI data centers. These centers are then used to train a Radial Basis Function Network (RBFN), enabling efficient classification of CQI data into distinct patterns. These patterns inform reinforcement learning-based scheduling decisions, which, in turn, achieve significant gains, up to 20% improvement, in fairness performance when scheduling multimedia consumers.**

***Index Terms*—CQI Compression, RRM, Scheduling, Clustering, Simulated Annealing, RBFN, Reinforcement Learning.**

I. Introduction

The rapid evolution of wireless networks is paving the way for the sixth generation (6G), enabling transformative applications like pervasive intelligence, holographic communications, and ultra-reliable low-latency automation. These ambitious use cases demand unprecedented performance from Radio Access Networks (RANs), including extreme data rates, ultra-low latency, and seamless technology integration. To address these challenges, 6G will adopt advanced technologies such as terahertz communications, intelligent surfaces, and AI-native architectures. Traditional Radio Resource Management (RRM) struggles to meet the demands of real-time scheduling in large-scale, heterogeneous 6G networks [1].

Open RAN plays a key role in introducing flexibility, modularity, and interoperability through the decoupling of hardware and software, enabling seamless integration of diverse technologies and scalable deployment of intelligent solutions for real-time optimization. Open RAN enables flexible RRM solutions given stringent Quality of Service (QoS) requirements and dynamic network environments. In this case, Machine Learning (ML) is emerging as a key enabler, leveraging historical data and real-time observations to optimize RRM functionalities. However, the scalability of ML is challenged by the large and variable dimensionality of data to be processed and ingested. Therefore, efficient data processing algorithms are essential for applying ML in Open RAN-enabled 6G systems [2].

Channel Quality Indicator (CQI) reports, which capture user channel conditions, are vital for ML-based scheduling and resource allocation but pose challenges due to high data complexity. This paper proposes a CQI compression framework for 6G networks, enabling efficient Reinforcement Learning (RL) in RRM [3], [4]. By simplifying the input space, RL benefits from faster and more accurate scheduling and resource allocation, meeting stringent QoS demands.

II. Related Work

Studies have explored various approaches to CQI compression to enhance RRM in LTE systems. For example, an adaptive threshold-based scheme to dynamically adjust CQI reporting is proposed in [5], reducing signaling overhead while maintaining throughput. In [6] the authors propose predictive filtering schemes for sub-band CQI feedback compression in LTE systems, significantly reducing signaling overhead while maintaining performance. These approaches highlight the role of efficient CQI compression in minimizing feedback while ensuring system reliability, although they focus on earlier LTE systems rather than more recent advancements.

Machine learning techniques have further improved CQI compression. For example, in [7] a Support Vector Machine (SVM)-based method for sub-band CQI feedback compression is proposed, showcasing the potential of supervised learning to reduce signaling overhead. Furthermore, in [8] a neural network model is implemented to estimate CQI in 5G systems, improving accuracy and efficiency in CQI assessment for improved resource allocation. These studies underscore the role of ML in addressing CQI compression challenges for efficient RRM in modern wireless systems.

CQI compression offers more than just signaling overhead reduction; it plays a vital role in data mining for RRM, especially in Open RAN. Compressed CQI can be used as input for ML models, to optimize scheduling and resource allocation. By transforming bandwidth and user dependent CQI data into

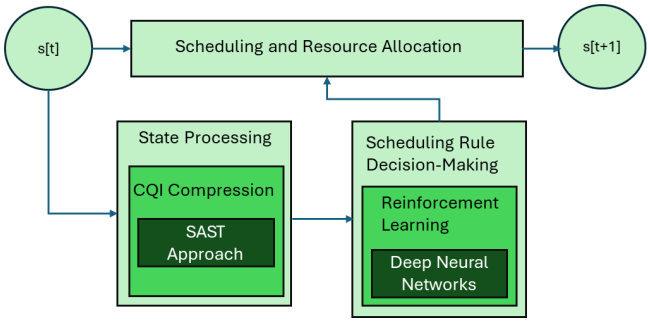

Fig. 1: System Model

compact representations will improve the intelligent decision-making in dynamic, multi-service networks.

III. System Model

The model in Fig. 1 illustrates a scheduling system in which, at each Transmission Time Interval (TTI) t, network observations from all active users—treated as states—are processed. Scheduling dynamically selects based on predefined rules which users get access to the resources at each TTI, while the allocation determines how those resources are distributed across the system. To maximize QoS provisioning, the scheduling rule is dynamically adapted at each TTI based on the scheduler state, with a RL approach leveraging neural networks to determine the most suitable rule. State processing—particularly the manipulation and compression of CQI reports—plays a pivotal role in enabling efficient data handling and informed decision-making.

A. Reinforcement Learning based Scheduling

At each TTI t, the set of active users is denoted as $\mathcal{I}_t = \{1, 2, ..., I_t\}$, where I_t is the maximum number of users. The system bandwidth is divided into J equal Resource Blocks (RBs), represented as $\mathcal{J} = \{1, 2, ..., J\}$. Assuming an error-free and delay-free uplink channel, each user $i \in \mathcal{I}_t$ reports their channel quality $x_{i,j}(t) = \{1, 2, ..., N\}$ on each RB $j \in \mathcal{J}$, where a higher value indicates better sub-channel quality. This information is essential for the scheduler in the base station's RRM to determine the data transmission capacity for each user at every TTI. However, sharing bandwidth among users is subject to constraints, turning resource allocation into a multi-objective optimization problem, expressed as:

$$\max_{a} \sum_i \sum_j a_{i,j}(t) \cdot \Gamma_i(\mathcal{Q}_i, x_{i,j}) \cdot \lambda_i(x_{i,j}), \quad (1)$$

$$s.t.$$

$$\sum_i a_{i,j}(t) \leq 1, \quad j = 1, 2, ..., J, \quad (1.a)$$

$$a_{i,j}(t) \in \{0, 1\}, \quad \forall i \in \mathcal{I}_t, \forall j \in \mathcal{J}, \quad (1.b)$$

$$\mathbf{q}_i \; constrained \; by \; \bar{\mathbf{q}}_i, \quad i = 1, 2, ..., I_t. \quad (1.c)$$

In (1), the decision variable $a_{i,j}(t) \in \{0, 1\}$ indicates whether RB $j \in \mathcal{J}$ is allocated to user $i \in \mathcal{I}_t$ ($a_{i,j} = 1$) or not ($a_{i,j} = 0$). The set of variables $\mathcal{Q}_i = \{\mathbf{q}_i, \bar{\mathbf{q}}_i\}$ represents the QoS indicators and their requirements, respectively, in terms of rate, delay, and packet loss. The function Γ_i represents a scheduling rule designed to meet specific QoS requirements, and $\lambda_i(x_{i,j})$ is the achievable rate of user i on RB j. The RL approach dynamically selects appropriate function Γ_i each TTI based on state $\mathbf{s}(t) \in \mathcal{S}$, ensuring QoS requirements $\bar{\mathbf{q}}_i$ are met for enhanced multimedia experiences. Here, state space processing—specifically, CQI compression—plays a critical role in achieving these objectives efficiently.

B. State Processing

The proposed CQI compression mechanism operates in three stages: pre-processing, classification, and optional statistical refinement. The pre-processing stage standardizes CQI data to be independent of bandwidth size, enabling consistent input handling. During classification, the mechanism utilizes an offline clustering process to identify CQI patterns and trains a Radial Basis Function Network (RBFN) based on these patterns. In online mode, the RBFN then classifies CQI patterns in real-time. Optionally, a statistical stage can further distill the classified patterns by emphasizing key features, particularly useful when a large number of output classes may hinder scheduling decisions in RL-based systems. If $\mathbf{x}_i(t) = [x_{i,1}, x_{i,2}, ..., x_{i,J}]$ is the CQI report at TTI t of user $i \in \mathcal{I}_t$, then the compressed CQI report becomes:

$$\mathbf{z}(t) = \mathcal{F}(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{I_t}), \quad (2)$$

where $\mathbf{z}$ is the compressed CQI of all users at TTI t, and $\mathcal{F}$ is the proposed distillation function encompassing the three stages. This approach streamlines CQI data for efficient use in ML-driven scheduling and resource allocation.

C. Simulated Annealing with Stochastic Tunneling

We propose a meta-heuristic approach leveraging a Simulated Annealing algorithm combined with a Stochastic Tunneling (SAST) function to enhance the CQI data mining process. This approach mitigates the risk of getting trapped in local minima during clustering and optimizes the training of the RBFN in the classification stage.

Given a current state of elements $\mathcal{B}$ and a candidate new state of elements $\mathcal{A}$, the acceptance probability of $\mathcal{A}$ from $\mathcal{B}$ is determined based on [9]:

$$\mathbb{P}(\mathcal{A}|\mathcal{B}) = min[1, \mathbb{F}(\mathcal{A}, \mathcal{B}, \mathbf{T})], \quad (3)$$

where $\mathbf{T}$ is the temperature, and $\mathbb{F}$ is the acceptance probability function with stochastic tunneling to enhance the quest of global optimum solutions, defined as [9]:

$$\mathbb{F}(\mathcal{A}, \mathcal{B}) = exp\left\{-[F(\mathcal{A}) - F(\mathcal{B})]/\mathbf{T}\right\}, \quad (4)$$

where $F : \mathcal{A} \times \mathcal{B} \rightarrow [0, 1]$ is the SAST function calculated based on:

$$F(\mathcal{X}) = 1 - exp\left\{-[f(\mathcal{X}) - f^*]/\omega\right\}. \quad (5)$$

Here, $\mathcal{X} \in \{\mathcal{A}, \mathcal{B}\}$, ω is the tunneling parameter, $f(\mathcal{A})$ and $f(\mathcal{B})$ are the energies of states $\mathcal{A}$ and $\mathcal{B}$, respectively, and f^* is the lowest energy discovered so far. In general, if $f(\mathcal{A}) < f(\mathcal{B})$, then $\mathcal{A}$ is a better option than $\mathcal{B}$. By employing the SAST function, the energy of each state is reported to the minimum value discovered so far during the optimization process. Therefore, if $F(\mathcal{A}) \leq F(\mathcal{B})$, then $\mathcal{A}$ is always accepted. Otherwise, $\mathcal{A}$ is accepted with probability $\mathbb{P}(\mathcal{A}|\mathcal{B})$ based on temperature $\mathbf{T}$. The principle of simulated annealing is to start with an initial temperature $\mathbf{T}_0$ which is decreased gradually. As this temperature decreases, the probability of accepting $\mathcal{A}$ from $\mathcal{B}$ also decreases when $F(\mathcal{A}) > F(\mathcal{B})$. The idea is to explore more at the beginning a broader neighborhood of $\mathcal{B}$ with good solutions, while drifting towards

more narrowed regions with low-energy as the temperature approaches to its minimum value. We propose to calculate the initial temperature as follows [9]:

$$\mathbf{T}_0 = \left\{ -\sum\nolimits_{d=1}^{L} [F_d(\mathcal{A}) - F_d(\mathcal{B})] \right\} \Big/ \Big[L \cdot \ln(\mathbb{P}_0) \Big] \quad (6)$$

where L is the number of iterations needed to approximate the average value of $F(\mathcal{A}) - F(\mathcal{B})$ and $\mathbb{P}_0$ is the initial acceptance probability. The temperature is reduced according to [9]:

$$\mathbf{T}_{new} = \mathbf{T}_{old} \cdot R_T, \quad (7)$$

where $R_T \in [0, 1]$ is the temperature reduction factor.

IV. Proposed Framework

A. Pre-processing Stage

To reduce dimensionality, we transform the original CQI report $\mathbf{x}_i$ of dimension J into a new vector $\mathbf{x}'_i$ of dimension N, where each $x'_{i,n}$ counts the occurrences of quality level $n \in \{1, 2, ..., N\}$ in $\mathbf{x}_i$. This transformation is given by:

$$x'_{i,n} = \frac{1}{J} \sum\nolimits_{\substack{j=1 \\ x_{i,j}=n}}^{J} x_{i,j}, \forall i \in \mathcal{I}_t, n = \{1, 2, ..., N\}, \quad (8.a)$$

$$\sum\nolimits_{n=1}^{N} x'_{i,n} = 1, \forall i \in \mathcal{I}_t. \quad (8.b)$$

Even though the dimensionality of $\mathbf{x}'_i$ is reduced (since $N \ll J$), values in $\mathbf{x}'_i$ are typically not uniformly distributed. To simplify computation further, we focus on the top M values in $\mathbf{x}'_i$ for each report, retaining the most significant CQI values.

For each $i \in \mathcal{I}_t$, we define two sets from $\mathbf{x}'_i$: - $\mathcal{M}_i = \{x'_{i,n,m}\}$, the set of top M data features where $m = 1, 2, ..., M$, and $\mathcal{R}_i = \{x'_{i,n,r}\}$, the residuals after determining the top M features, where $r = 1, 2, ..., N - M$. To ensure that the transformed vector maintains linearity as per (8.b), the residuals are assigned to the top m features. The assignment of elements from $\mathcal{R}_i$ to $\mathcal{M}_i$ minimizes the index difference in $\mathbf{x}'_i$. This results in refined residual sets $\mathcal{R}_{i,m} = \{x'_{i,n,r(m)}\}$, where each residual $x'_{i,n,r(m)}$ is matched to a top feature $x'_{i,n,m}$. Here, $m = 1, 2, \ldots, M$ and $r(m) = 1, 2, \ldots, R(m)$, with $R(m)$ representing the number of residual features assigned to top feature m. This process is formalized as:

$$x'_{i,n,p} = \begin{cases} x'_{i,n,m} + \sum_{r(m)}^{R(m)} x'_{i,n,r(m)}, & m \leq M, \\ 0, & otherwise. \end{cases} \quad (9)$$

Thus, we obtain the pre-processed CQI vector from $\mathbf{x}_i = [x_{i,j}]$, to $\mathbf{x}'_{i,M} = [x'_{i,n,p}]$, where $\{n, p\} = 1, 2, \ldots, N$ with M being the number of top data features from $\mathbf{x}'_i$.

B. Classification Stage

In this approach, CQI data is gathered from multiple active users $i \in \mathcal{I}_t$ over an extended time horizon across TTIs t. We denote by $\mathcal{X}_M$ the processed collection of CQI data points from all users, with each data point represented by its top M most significant features. The goal is to identify patterns within the collection of CQI data points $\mathcal{X}_M$, and subsequently classify new CQI data according to these patterns. The classification process involves two levels: offline and online. In offline mode, clustering is first applied to uncover patterns in historical data, followed by RBFN training to generalize these patterns for classification. In the online mode, the trained RBFN classifies incoming CQI reports in real-time, assigning them to the identified patterns. This dual-stage framework enables offline classifier training while supporting rapid and accurate categorization of CQI data in online settings based on established patterns.

1) Clustering: addresses the location-based problem of identifying the most suitable data centers for organizing the collected data $\mathcal{X}_M$. Let us define the set of centers as $\mathcal{C}_M = \{\mathbf{c}_k \mid k = 1, 2, \ldots, K\}$, where each center is represented by $\mathbf{c}_k = \{c_{1,k}, c_{2,k}, \ldots, c_{N,k}\}$, and K denotes the total number of centers. This set $\mathcal{C}_M$ is derived from the collected data parameterized by M; however, the resulting centers may not satisfy the condition specified in (9). The goal of clustering is to determine the optimal set of centers $\mathcal{C}^*_M$ that minimizes the average distortion, typically measured as the mean squared distance, and calculated as:

$$d(\mathcal{C}_M) = 1/X_M \cdot \sum\nolimits_{u=1}^{X_M} ||\mathbf{x}'_{u,M} - \mathbf{c}_{k(u)}||^2, \quad (10)$$

where $\mathbf{c}_{k(u)}$ denotes the centroid of the cluster to which the data point $\mathbf{x}'_{u,M}$ is assigned, and $||\cdot||^2$ represents the squared Euclidean distance.

The clustering algorithms work in iterations $l = 1, 2, ..., L$, where L is the maximum number of iterations. If $\mathcal{C}^l_M$ is the set of processed CQI centers computed up to and including iteration l, then the goal of any clustering algorithm is to find the optimal set of centers $\mathcal{C}^*_M$ within the maximum number of iterations, such that the distortion $d(\mathcal{C}^*_M)$ is minimized. Two types of algorithms are studied to deal with this objective:

a) K-means: at each iteration, the neighborhood $\mathcal{V}^l_k$ is determined for each center k, where $\mathcal{V}^l_k$ defines the sub-set of data points for which $\mathbf{c}^l_k$ is the nearest neighbor; then, the weighted centroid of each neighbor becomes:

$$\bar{\mathbf{c}}^l_k = 1/|\mathcal{V}^l_k| \cdot \sum\nolimits_{v=1}^{|\mathcal{V}^l_k|} \mathbf{x}'_{v,M}, \quad (11)$$

where, $|\mathcal{V}^l_k|$ is the number of data points within the neighborhood $\mathcal{V}^l_k$; next, each point in the collection $\mathcal{X}_M$ is assigned to its corresponding new centroid, resulting in an updated set of centers denoted by $\mathcal{C}^{l+1}_M = \{\bar{\mathbf{c}}^l_1, \bar{\mathbf{c}}^l_2, \ldots, \bar{\mathbf{c}}^l_K\}$. These steps are repeated for the number of iterations L, until optimal $\mathcal{C}^*_M$ is found. To speed-up the neighborhood search in each iteration, the collected data $\mathcal{X}_M$ is stored in a kd-tree [10], and for each new calculated centroid, the neighbors are automatically given. K-means is susceptible to converging to local minima, and this paper proposes methods to address this challenge.

b) Swap Heuristic: Instead of explicitly determining neighborhoods and centroids, this algorithm randomly selects a data sample from $\mathcal{X}_M$ as the new center at each iteration and continues this process until the specified number of iterations is reached. This approach helps avoid local minima but may take more time to reduce average distortion.

To enable the use of the proposed meta-heuristic algorithm in clustering, we divide the original number of iterations L in E epochs, where $e = 1, 2, ..., E$ is the epoch index with L_e iterations. Each epoch starts with the swap heuristic clustering and then continues with k-means until the number of iterations L_e per epoch is achieved in order to minimize the average distortion from (10). To minimize distortion and avoid the convergence to suboptimal minima, the set of initial centers $\mathcal{C}^0_M$ at the beginning of each epoch becomes important. In

Algorithm I: SAST for Clustering

input parameters $\{M, \mathcal{X}_M, K, E, L_e, \mathbf{T}_0\}$
for each $e \in \{1, 2, ..., E\}$
  for each $l \in \{1, 2, ..., L_e\}$
    if $(l = 1)$
      randomize one center $\forall \mathbf{c}_k^1 \in \mathcal{X}_M$, $k = 1, .., K$
    else
      determine neighborhood $\mathcal{V}_k$ for each $\mathbf{c}_k^l \in \mathcal{C}_M^l$
      calculate centroid $\bar{\mathbf{c}}_k^l$ based on (11)
      assign $\mathbf{c}_k^{l+1} = \bar{\mathbf{c}}_k^l$
    assign each point $\mathbf{x}'_{i,M} \in \mathcal{X}_M$ to closest $\mathbf{c}_k^l \in \mathcal{C}_M^l$
    calculate distortion $d(\mathcal{C}_M^l)$ based on (10)
    if $d(\mathcal{C}_M^l) < d^*$ **then** $\mathcal{C}_M^* = \mathcal{C}_M^l$
    if $(l = L_e)$
      if $d(\mathcal{C}_M^{L_e}) < d(\mathcal{C}_M^0)$ **then** $\mathcal{C}_M^0 = \mathcal{C}_M^{L_e}$
      else
        calculate $\mathbb{P}(\mathcal{C}_M^{L_e}|\mathcal{C}_M^0)$ based on (3), (4), (5)
        if $\mathbb{P}(\mathcal{C}_M^{L_e}|\mathcal{C}_M^0) \geq \mathbb{P}_a$, then $\mathcal{C}_M^0 = \mathcal{C}_M^{L_e}$
        else keep $\mathcal{C}_M^0$
      decrease temperature $\mathbf{T}_{new}$ based on (6), (7)
  end iteration
end epoch
return $\mathcal{C}_M^*$

normal way, $\mathcal{C}_M^0$ should be the best set of centers discovered so far in the previous epochs. But as seen, this leads to local minima solutions. To avoid this, we employ a solution that accepts non-better solutions from the previous epoch based on the simulated annealing approach introduced above. If $\mathcal{A} = \mathcal{C}_M^{L_e}$ and $\mathcal{B} = \mathcal{C}_M^0$ are the sets of centers at end and beginning of epoch e, respectively, then the initial set of centers at the beginning of the next epoch is decided based on:

$$\mathcal{C}_M^0 = \begin{cases} \mathcal{C}_M^{L_e}, & d(\mathcal{C}_M^{L_e}) < d(\mathcal{C}_M^0), \\ \mathcal{C}_M^{L_e}, & d(\mathcal{C}_M^{L_e}) \geq d(\mathcal{C}_M^0) \;\&\; \mathbb{P}(\mathcal{C}_M^{L_e}|\mathcal{C}_M^0) \geq \mathbb{P}_a, \\ \mathcal{C}_M^0, & d(\mathcal{C}_M^{L_e}) \geq d(\mathcal{C}_M^0) \;\&\; \mathbb{P}(\mathcal{C}_M^{L_e}|\mathcal{C}_M^0) < \mathbb{P}_a, \end{cases} \quad (12)$$

where $\mathbb{P}(\mathcal{C}_M^{L_e}|\mathcal{C}_M^0)$ is the acceptance probability of $\mathcal{C}_M^{L_e}$ as initial set when the current initial set of centers is $\mathcal{C}_M^0$ and $\mathbb{P}_a$ is the probability threshold. At the beginning, the temperature is high and the non-better set of centers have higher probability to be accepted as the initial centers at the beginning of the next epoch. As the temperature gradually decreases, better solutions have a greater probability of acceptance. It is worth mentioning that the algorithm keeps track of optimal set $\mathcal{C}_M^*$ that minimizes distortion in each epoch. Algorithm I presents the proposed meta-heuristic clustering approach.

2) Classification: aims to create an automatic prediction model able to classify each preprocessed observation $\mathbf{x}'_{i,M}$ of user $i \in \mathcal{I}_t$ in the corresponding pattern. We define $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_K\}$ the set of encoded patterns, where $\mathbf{p}_k = [p_{k,1}, p_{k,2}, ..., p_{k,O}]$ with O being the total number of bits needed to encode K, with $p_{k,o} \in \{-1, 1\}$, $o = 1, 2, ..., O$. The proposed classification function maps $\mathcal{X}_M$ in a finite set of patterns $\mathcal{P}$ based on the following labeled set:

$$\mathcal{G} = \{(\mathbf{x}'_{u,M}, \mathbf{p}_u)|\mathbf{x}'_{u,M} \in \mathcal{X}_M, \mathbf{p}_u \in \mathcal{P}, u = \overline{1, X_M}\}. \quad (13)$$

Given $\mathcal{G}$, let Φ be such a classification function that takes the form of a RBF function:

$$\Phi(\mathbf{x}'_{u,M}) = \Phi_2[\mathbf{W} \times \Phi_1(\mathbf{x}'_{u,M})] = \mathbf{p}_u, \quad (14)$$

where, $\mathbf{W} = \{w_{k,o}|k = 1, 2, ..., K; o = 1, 2, ..., O\}$ is the matrix of weights connecting the hidden and output layers, and $\Phi_1 = [\phi_{u,1}^{(1)}, \phi_{u,2}^{(1)}, ..., \phi_{u,K}^{(1)}]$ is the vector of RBF functions of the the hidden layer. Among multiple representations, the RBF function $\phi_{u,k}^{(1)} : [0, 1]^N \to [-1, 1]$ for each hidden node k takes the Gaussian form given by:

$$\phi_{u,k}^{(1)} = exp\left(- \frac{||\mathbf{x}'_{u,M} - \mathbf{c}_k||^2}{2\sigma^2} \right), \quad (15)$$

where $\sigma > 0$ is the Gaussian parameter that is a prori tuned. Finally, $\Phi_2 = [\phi_1^{(2)}, ..., \phi_O^{(2)}]$ is the vector of non-linear transformations, taking the form of tangent hyperbolic.

In practice, a perfect interpolation in (14) is difficult to achieve since a complete set of $\mathcal{X}_M$ with all possible combinations of the pre-processed CQI reports is complicate to be found, especially for $M > 2$. Therefore, we can train the vector of weights $\mathbf{W}$ to obtain a good approximation of classification function $\widetilde{\Phi}$, represented as:

$$\widetilde{\Phi}(\mathbf{x}'_{u,M}) = \Phi(\mathbf{x}'_{u,M}) - \Delta(\mathcal{W}, \mathbf{x}'_{u,M}), \quad (16)$$

where Δ is the loss or the error function. Then, the main objective of the classification stage would be to minimize the loss function by training the weights $\mathbf{W}$ without getting stuck into local minima solutions. In this sense, we keep $\mathcal{G}$ as a validation set and get a training data set $\mathcal{H}$ that can be collected from the network and defined as:

$$\mathcal{H} = \{(\mathbf{x}'_{i,M}, \mathbf{p}_i)|\mathbf{x}'_{i,M}, \mathbf{p}_i \in \mathcal{P}, i = \overline{1, I_t}\}. \quad (17)$$

We train the RBFN similar to the clustering algorithms by using E number of epochs and L_e iterations, where $e = 1, 2, .., E$. Since the RBFN training is coupled to the network, an epoch is represented by all CQIs received in one TTI. In each iteration $l = 1, 2, ..., L_e$, a data sample $(\mathbf{x}'_M, \mathbf{p})$ is selected from $\mathcal{H}$ or $\mathcal{G}$, and the set of $\mathcal{W}_l$ is updated based on error back-propagation. To enhance generalization and avoid local minima solution, we employ a decision-making algorithm based on the same SAST principle to pick up samples from training or validation sets at each iteration. Based on the same principle, the initial set of weights $\mathcal{W}_0$ at the beginning of each epoch is selected to find optimal weights $\mathcal{W}^*$ and minimize error Δ. Therefore, three processes are followed when training this structure:

a) Sample Feed-Forward: each epoch starts with samples from $\mathcal{H}$ and continuously monitor when a data sample from $\mathcal{G}$ should be forwarded in the RBFN. Let us consider $\mathcal{A} = \mathcal{W}^l$ and $\mathcal{B} = \mathcal{W}^{l-1}$ as the weight sets updated in iterations l and $l - 1$, respectively. Validation samples are preferred starting with iteration $l + 1$, if the next condition is met:

$$(\mathbf{x}'_M, \mathbf{p}) \in \begin{cases} \mathcal{H}, & \Delta(\mathcal{W}^l) < \Delta(\mathcal{W}^{l-1}), \\ \mathcal{H}, & \Delta(\mathcal{W}^l) \geq \Delta(\mathcal{W}^{l-1}) \;\&\; \mathbb{P} \geq \mathbb{P}_a, \\ \mathcal{G}, & \Delta(\mathcal{W}^l) \geq \Delta(\mathcal{W}^{l-1}) \;\&\; \mathbb{P} < \mathbb{P}_a, \end{cases} \quad (18)$$

where $\mathbb{P} = \mathbb{P}(\mathcal{W}^l|\mathcal{W}^{l-1})$ is the acceptance probability of $\mathcal{W}^l$ when the previous set of weights is $\mathcal{W}^{l-1}$ based on data samples from $\mathcal{H}$.

b) Error Back-Propagation: in each iteration $l = 1, 2, ..., L_e$, the error $\Delta(\mathcal{W}^l)$ is calculated based on the RBFN response $\hat{\mathbf{p}}$ and the real pattern $\mathbf{p}$ and is reinforced to update the set of weights based on the gradient descent principle:

$$\mathbf{W}^l = \mathbf{W}^{l-1} + \eta \cdot \mathbf{x}'_M \times \Delta, \quad (19)$$

Algorithm II: SAST for RBFN Training

for each $e(TTI\ t) \in \{1, 2, ..., E\}$
  for each $l \in \{1, 2, ..., L_e\}$
    get a CQI report $\mathbf{x}_i$
    preprocess $\mathbf{x}_i$ to $\mathbf{x}'_{i,M}$ based on (8.a), (8.b) and (9)
    assign pattern $\mathbf{p}_i$ based on closest neighborhood and $\mathcal{C}_M$
    feed-forward $\mathbf{x}'_{i,M}$ from $\mathcal{H}$ or $\mathcal{G}$
    calculate error Δ based on $\mathbf{p}$ and $\hat{\mathbf{p}}$
    back-propagate Δ and update $\mathcal{W}^{l-1}$ to $\mathcal{W}^{l}$ based on (19)
    calculate $\mathbb{P}(\mathcal{W}^{l}|\mathcal{W}^{l-1})$ based on (3), (4), (5)
    determine next sample based on (18)
    reduce temperature inner SAST based on (7)
    if $l = L_e$
      calculate $\mathbb{P}(\mathcal{W}^{L_e}|\mathcal{W}^{0})$ based on (3), (4), (5)
      determine next $\mathcal{W}^0$ based on (20)
      reduce temperature outer SAST based on (7)
    end if
  end iteration l
end epoch e

where $\eta \in [0, 1]$ is the learning rate that sets the learning speed of the RBF network.

c) Acceptance of Weights: in each iteration, we keep track of $\mathcal{W}^*$, corresponding to the lowest error observed so far. To prevent the solution from getting trapped in a local minimum, we adopt a similar acceptance strategy at the beginning of each epoch, as described in (12). Consequently, the initial set of weights $\mathcal{W}^0$ is set according to the following principle:

$$\mathcal{W}^0 = \begin{cases} \mathcal{W}^{L_e}, & \Delta(\mathcal{W}^{L_e}) < \Delta(\mathcal{W}^0), \\ \mathcal{W}^{L_e}, & \Delta(\mathcal{W}^{L_e}) \geq \Delta(\mathcal{W}^0)\ \&\ \mathbb{P}(\mathcal{W}^{L_e}|\mathcal{W}^0) \geq \mathbb{P}_a, \\ \mathcal{W}^0, & \Delta(\mathcal{W}^{L_e}) \geq \Delta(\mathcal{W}^0)\ \&\ \mathbb{P}(\mathcal{W}^{L_e}|\mathcal{W}^0) < \mathbb{P}_a. \end{cases} \tag{20}$$

Since this optimization targets two decisions, we denote two loops of SAST algorithm: *a)* the inner SAST, which involves selecting the data sample to be input into the RBFN and resets at each TTI; and *b)* the outer SAST, which determines the initial set of weights and spans the entire training session of the RBFN structure. This is outlined in Algorithm II.

In real-time networking conditions, the trained RBFN operates at each TTI t, producing a set $\mathcal{P}(t) = \{\hat{\mathbf{p}}_i, i = 1, 2, ..., I_t\}$, where each $\hat{\mathbf{p}}_i = [\hat{p}_{i,1}, \hat{p}_{i,2}, ..., \hat{p}_{i,O}]$ represents the encoded pattern output by the RBFN for user $i \in \mathcal{I}_t$. Decoding $\hat{\mathbf{p}}_i$ reveals the cluster index k to which the preprocessed CQI sample $\mathbf{x}'_{i,M}$ belongs. Clusters are indexed from 1 to K, where cluster 1 corresponds to the worst channel conditions and cluster K corresponds to the best. At each TTI, the number of appearances of each cluster k is denoted as y_k, and the resulting classification output is represented by:

$$\mathbf{y}(t) = [y_1, y_2, ..., y_K]. \tag{21}$$

C. Statistical Stage

Integrating the classification vector $\mathbf{y}$ into RL-based scheduling presents challenges due to its dependence on the number of clusters K, which can vary across scenarios. A high K can complicate the RL-based scheduler's ability to effectively process the classification vector, potentially overlooking other critical data features. Thus, we process $\mathbf{y}$ using statistical functions to reduce it to a fixed-dimensional representation, that remains independent of K. We focus on four key features derived from $\mathbf{y}$, such as: *a)* n_a, the number of active clusters in $\mathbf{y}$, *b)* σ_a, the dispersion of active clusters in $\mathbf{y}$, c) d_y, the minimum Euclidean distance between $\mathbf{y}$ and support classification vectors $\bar{\mathbf{y}}_k$ for $k = 1, 2, ..., K$ where $\bar{\mathbf{y}}_k$ has 1 at position k and 0 elsewhere; d) k, the index that minimizes the distance in c). By transforming $\mathbf{y}$ into these features, we provide a consistent and manageable input for RL-based schedulers, enabling effective decision-making regardless of K, and producing the final compact representation at each TTI t:

$$\mathbf{z}(t) = [n_a, \sigma_a, d_k, k]. \tag{22}$$

V. Simulation Results

We analyze downlink transmissions for $I = 1000$ users within a 20 MHz bandwidth ($J = 100$) in a microcell urban environment, modeled using the Jakes Model with 12 fading paths. Users travel at a speed of 120 km/h in random directions to enhance the diversity of CQI reports. Each CQI report contains $N = 15$ quality levels and is collected every 1ms across 19 cells. For data collection, the top $M = \{3, 4, 5\}$ CQI elements are extracted, with the process terminating once no new data points are detected. This results in datasets of sizes $X_3 = 33596$, $X_4 = 144179$ and $X_5 = 206473$. These datasets are then used to train clustering algorithms and RBFN to classify CQI reports into distinct patterns. The trained models are subsequently applied in real-time to compress CQI reports and generate input states for RL-based scheduling algorithms, aiming to optimize fairness in multimedia service delivery. Network simulations and data collection were carried out using the simulator from [11], which is an extended version of [12], enhanced with data processing and RL algorithms.

A. SAST-based Clustering

Two clustering settings with $K = \{64, 512\}$ centers are analyzed, employing iterative k-means with kd-tree, a swap heuristic, and the proposed SAST-based clustering method. Additionally, a simulated annealing approach without stochastic tunneling, as described in [9], is evaluated. Each clustering method is tested over $E = 100$ epochs, with $L_e = 10$ iterations per epoch. For the simulated annealing approach, the parameters include a tunneling factor $\omega = 0.02$, an initial acceptance probability $\mathbb{P}_0 = 0.5$, and a temperature reduction factor $R_T = 0.95$. Table I provides a performance comparison of the clustering algorithms based on average distortion and CPU time required to compute the sets of centers $\mathcal{C}_3$, $\mathcal{C}_4$ and $\mathcal{C}_5$. Among the evaluated methods, the iterative k-means algorithm is the most computationally intensive, whereas the swap heuristic yields the highest distortion. By strategically accepting non-better solutions, clustering algorithms utilizing the simulated annealing approach achieve lower average distortion across all scenarios. The inclusion of a tunneling function in the simulated annealing-based method enhances both distortion and computational complexity compared to its initial implementation [9]. The SAST-based clustering method demonstrates the most significant distortion reduction for $M = 3$, followed by $M = 4$ and $M = 5$. The center sets obtained from the SAST algorithm will subsequently be applied in RBFN for further processing.

TABLE I: Performance Comparison of Clustering Techniques (Distortion and CPU Time)

Method	M = 3		M = 4		M = 5	
	K = 64	K = 512	K = 64	K = 512	K = 64	K = 512
k-means	0.0193 / 16.09	0.0022 / 75.01	0.0111 / 56.90	0.0027 / 223.80	0.0084 / 137.19	0.0030 / 504.21
swap	28.1% / -1.2%	54.3% / -2.2%	31.2% / -4.1%	32.5% / -3.5%	26.4% / -3.1%	27.5% / -2.7%
sa	-7.1% / -3.3%	-5.7% / -3.9%	-2.4% / -3.8%	-3.5% / -3.0%	-1.5% / -5.7%	-2.8% / -1.6%
sast	-7.3% / -3.4%	-5.9% / -4.0%	-2.6% / -6.8%	-3.9% / -2.2%	-1.8% / -6.8%	-3.3% / -1.9%

TABLE II: RBFN Hyperparameters and Best Mean Error for Different Configurations

Params	M=3 K=64	M=3 K=512	M=4 K=64	M=4 K=512	M=5 K=64	M=5 K=512
η	0.089	0.022	0.050	0.006	0.032	0.002
σ	50	440	90	370	120	310
$\Delta(\mathcal{G})$	0.028	0.037	0.034	0.115	0.073	0.169
$\Delta(\mathcal{HG})$	0.01	0.048	0.042	0.152	0.091	0.228

B. SAST-based RBFN Training

The RBFN weights $\mathcal{W}$ are trained using the sets of data centers $\mathcal{C}_3$, $\mathcal{C}_4$ and $\mathcal{C}_5$ with $K = \{64, 512\}$, which are derived from the SAST-based clustering algorithm. Since the collected datasets $\mathcal{X}_3$, $\mathcal{X}_4$ and $\mathcal{X}_5$ cannot fully represent all possible combinations in the CQI reports, it is critical to train the RBFN effectively to enhance generalization. This ensures that new, previously unseen samples are classified into the correct pattern classes. To achieve this, the dataset $\mathcal{G}$ is reserved for validation, while the training set $\mathcal{H}$ is dynamically linked to the network simulator. However, as the CQI data in $\mathcal{H}$ may lack diversity in the short term, there is a risk of overfitting the RBFN. To mitigate this, the SAST principle is applied at each iteration to select training samples alternately from $\mathcal{H}$ and $\mathcal{G}$. Furthermore, SAST is employed at the end of each training epoch to decide whether to accept non-better weights. This allows the algorithm to explore better solutions in subsequent iterations, ultimately improving the RBFN's robustness and generalization capabilities.

We perform 4-fold cross-validation on the validation set to determine the optimal RBFN hyperparameters, including the learning rate (η) and Gaussian parameter (σ), by minimizing the error (Δ). The results are presented in Table II. During RBFN training, we replicate the same network conditions used during data collection, setting the number of epochs to 100.000 TTIs and the number of iterations per epoch to $L_e = 1000$, equal to the number of active users in the network. The SAST-based algorithms are applied consistently for both inner and outer loops, as in the clustering stage. At the end of training, we compare the mean error (Δ) between two approaches: training the RBFN solely on $\mathcal{G}$ without SAST ($\Delta(\mathcal{G})$) and training the RBFN with both inner and outer SAST algorithms ($\Delta(\mathcal{HG})$). As shown in Table II, the SAST-based approach yields a slightly higher training error compared to the traditional method. However, this trade-off promotes better generalization and results in improved weight configurations, enabling the RBFN to more effectively classify new CQI observations into the desired patterns. The trained RBFN models for all configurations ($M = \{3, 4, 5\}$ and $K = \{64, 512\}$) are then used to compute compressed states $\mathbf{z}$. These states serve as inputs for various RL algorithms to enhance user fairness in scheduling multimedia content on the 6G radio interface.

C. Impact on Scheduling Performance

To evaluate the impact of the proposed CQI compression approach on scheduling, we use the fairness criterion defined by the Next Generation Mobile Networks (NGMN) alliance, as detailed in [13]. This criterion requires the Cumulative Distribution Function (CDF) of user average throughput at each TTI to fall within a specified region, as illustrated in Fig. 2. Figure 2.a demonstrates cases of unfairness and overfairness: in the unfair scenario, users with favorable CQI conditions are prioritized, maximizing overall throughput at the expense of fairness, while the overfair scenario excessively prioritizes fairness, leading to inefficient resource usage. The desirable case, shown in Fig. 2.b, positions the CDF within the feasible area, ensuring that system throughput is maximized while meeting the NGMN fairness criterion.

The primary objective of the RL-based scheduler is to adjust the scheduling decisions at each TTI to ensure that the user throughput CDF remains within the feasible region, achieving both fairness and optimal resource utilization. This can also be achieved by extending the basic parametrization of the Proportional Fair (PF) scheduling rule from [13]:

$$\Gamma_i(\lambda_i, x_{i,j}) = \lambda_i(x_{i,j})^{\beta_t - 1} / \Lambda_i^{\alpha_t} \tag{23}$$

where Λ_i is the average user throughput used to calculate CDF. Various RL algorithms [14] (Q-learning, SARSA, QV-learning, ACLA, CACLA) are trained to select the optimal scheduling parameters $[\alpha_t, \beta_t]$ at each TTI to maximize a reward function aligned with NGMN fairness requirements. The training is conducted under two state definitions: *a)* $\mathbf{s} \in \mathcal{S}$, including Λ_i statistics and the compressed CQI state $\mathbf{z}$, as defined in (22); and *b)* $\mathbf{s} \in \mathcal{S}$ with Λ_i statistics but without $\mathbf{z}$.

The RL algorithms are trained using the network settings described in [14] for scheduling multimedia services with full-buffer traffic. Their performance is evaluated across 10 different network configurations, and the results are averaged. Table III presents the mean percentage of TTIs where the scheduler operates in the $\mathcal{UF}$ (unfair) and $\mathcal{FS}$ (feasible) regions for various RL algorithms, state-of-the-art approaches (AS, PF) [14], and different CQI compression configurations. The results show that without the compressed CQI state, RL algorithms fail to surpass the state-of-the-art AS algorithm. However, when the compressed CQI state $\mathbf{z}$ is included in the RL state $\mathbf{s}$, performance generally improves with higher values of M and K. The best results are achieved by the CACLA algorithm, which operates in the $\mathcal{FS}$ region for over 97% of

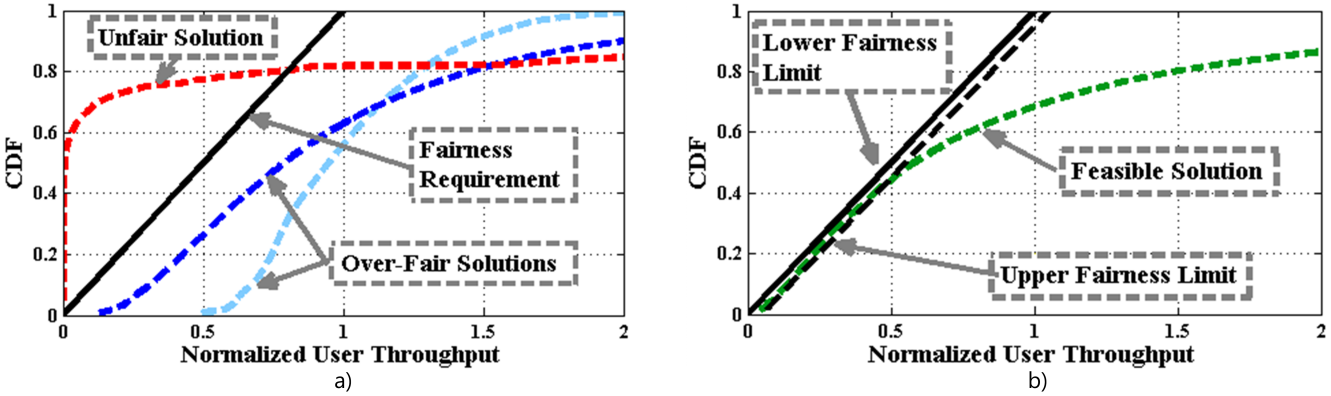

Fig. 2: NGMN Fairness Requirement: a) unfair and over-fair cases; b) feasible case

TABLE III: Performance Comparison of RL-based Scheduling Approaches in terms of Mean Percentage of TTIs when the CDF is Located in the Unfair ($\mathcal{UF}$) / Feasible ($\mathcal{FS}$) Areas

Algorithm	No Compression	M=3, K=64	M=3, K=512	M=4, K=64	M=4, K=512	M=5, K=64	M=5, K=512
PF	1.1 / 0.5	1.1 / 0.5	1.1 / 0.5	1.1 / 0.5	1.1 / 0.5	1.1 / 0.5	1.1 / 0.5
AS	10.7 / 85.9	10.7 / 85.9	10.7 / 85.9	10.7 / 85.9	10.7 / 85.9	10.7 / 85.9	10.7 / 85.9
Q	1.9 / 13.0	11.0 / 86.7	25.8 / 72.5	2.8 / 76.4	12.7 / 77.9	6.3 / 80.0	2.8 / 86.2
SARSA	2.9 / 69.7	1.9 / 88.0	32.2 / 58.1	2.5 / 90.8	1.7 / 83.7	2.5 / 92.8	8.1 / 89.8
QV	11.4 / 80.3	3.4 / 90.5	8.5 / 91.1	5.9 / 92.0	5.3 / 89.7	6.1 / 92.5	1.7 / 95.6
ACLA	9.9 / 83.0	4.6 / 90.9	4.1 / 94.2	3.3 / 94.5	4.2 / 92.1	5.0 / 93.6	7.3 / 91.5
CACLA	16.9 / 82.7	2.1 / 96.8	1.8 / 96.5	1.7 / 96.7	1.7 / 97.8	1.7 / 97.6	1.8 / 97.4

TTIs with $M = 4$ and $K = 512$. Additionally, increasing M and K yields up to a 2% performance gain, indicating that even the lowest configuration ($M = 3, K = 64$) offers a practical and efficient option for deployment.

VI. Conclusion

This paper introduces a novel compression mechanism designed to reduce the state space complexity arising from large CQI reports in 6G networks. The proposed approach leverages simulated annealing optimization with a stochastic tunneling function to enhance the search for optimal solutions and improve the classification of CQI data. A clustering-based method is employed to identify the best set of CQI centers, and RBFN structures are trained for various configurations of the compression scheme to generate a compressed state that retains the most valuable information. When applied to fairness-oriented scheduling using reinforcement learning, the results demonstrate significant performance improvements. Incorporating the compressed CQI state into the RL algorithm's state representation achieves gains of up to 20%, showcasing the effectiveness of the proposed approach in improving resource allocation and fairness in 6G networks.

Acknowledgment

This research was conducted as part of the UKIERI-SPARC project "DigIT—Digital Twins for Integrated Transportation Platform", grant number UKIERI-SPARC/01/23.

References

[1] D. B. da Costa, Q. Zhao, M. Chafii, F. Bader, and M. Debbah, *6G: Vision, Applications, and Challenges*. Springer International Publishing, 2024, pp. 15–69.

[2] M. Polese, M. Dohler, F. Dressler, M. Erol-Kantarci, and R. J. R. Knopp, "Empowering the 6G Cellular Architecture with Open RAN," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 245 – 262, 2023.

[3] I. S. Comșa, M. Aydin, S. Zhang, P. Kuonen, and J.-F. Wagen, "Reinforcement Learning based Radio Resource Scheduling in LTE-advanced," in *17th International Conference on Automation and Computing*, November 2011, pp. 219 – 224.

[4] I. S. Comsa, M. Aydin, S. Zhang, P. Kuonen, and J. Wagen, "Multi Objective Resource Scheduling in LTE Networks Using Reinforcement Learning," *International Journal of Distributed Systems and Technologies (IJDST)*, vol. 3, no. 2, pp. 1 – 19, 2012.

[5] M. Q. Abdulhasan, M. I. Salman, C. K. Ng, N. K. Noordin, S. J. Hashim, and F. Hashim, "An Adaptive Threshold Feedback Compression Scheme Based on Channel Quality Indicator (CQI) in Long Term Evolution (LTE) System," *Wireless Personal Communications*, vol. 82, pp. 2323 – 2349, Feb. 2015.

[6] M. Cordina and C. Debono, "Robust Predictive Filtering Schemes for Sub-band CQI Feedback Compression in 3GPP LTE Systems," *IET Communications*, vol. 11, no. 11, pp. 1797 – 1807, May 2017.

[7] M. Cordina and C. J. Debono, "A Support Vector Machine based Sub-band CQI Feedback Compression Scheme for 3GPP LTE Systems," in *Int. Symp. on Wireless Comm. Systems*, 2017, pp. 325–330.

[8] S. K. Vankayala and K. G. Shenoy, "A Neural Network for Estimating CQI in 5G Communication Systems," in *IEEE Wireless Communications and Networking Conference Workshops*, June 2020, pp. 1 – 5.

[9] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "A Local Search Approximation Algorithm for K-means Clustering," in *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, 2002, pp. 10 – 18.

[10] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An Efficient K-means Clustering Algorithm: Analysis and Implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[11] I.-S. Comșa, *Sustainable Scheduling Policies for Radio Access Networks Based on LTE Technology*. Ph.D. dissertation, School Comput. Sci. Technol., Univ. of Bedfordshire, Luton, U.K., 2014.

[12] G. Piro, L. A. Grieco, G. Boggia, F. Capozzi, and P. Camarda, "Simulating LTE Cellular Systems: An Open-Source Framework," *IEEE Trans. on Vehicular Nets.*, vol. 60, no. 2, pp. 498 – 513, 2011.

[13] I.-S. Comșa, M. Aydin, S. Zhang, P. Kuonen, J.-F. Wagen, and Y. Lu, "Scheduling Policies Based on Dynamic Throughput and Fairness Tradeoff Control in LTE-A Networks," in *IEEE Local Computer Networks (LCN)*, September 2014, pp. 418–421.

[14] I.-S. Comșa, S. Zhang, M. Aydin, P. Kuonen, R. Trestian, and G. Ghinea, "A Comparison of Reinforcement Learning Algorithms in Fairness-Oriented OFDMA Schedulers," *Information*, vol. 10, no. 10, 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/10/315