Open RAN Conflict Agents: Detecting and Mitigating xApp Conflicts with Generative Agents

Dae Cheol Kwon, Xinyu Zhang
Department of Electrical and Computer Engineering, University of California San Diego, San Diego, USA
Emails:{dckwon, xyzhang}@ucsd.edu

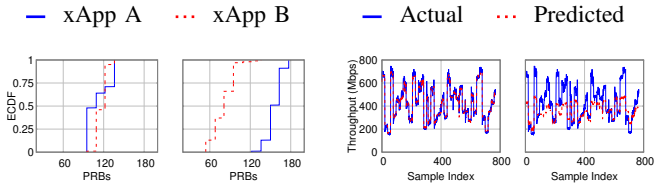
***Abstract*—The Open Radio Access Network (ORAN) architecture promotes flexibility and innovation by allowing independently developed xApps and rApps to control network behavior. However, this disaggregated design also introduces risks of resource conflicts that are difficult to detect and resolve, especially in unseen or dynamic environments. We propose ORCA, a conflict management framework that leverages generative LLM agents to reason about the importance of control parameters under environmental uncertainty. The framework incorporates symbolic reasoning through a novel β-Reasoning Cascade and introduces Segmented Active RAG (SA-RAG) to retrieve relevant domain knowledge dynamically. Evaluations on diverse environments show that ORCA reduces cosine error by 60.7% and NRMSE by 65.1% over the best baselines, achieving the most accurate feature importance estimates for conflict detection. Our findings demonstrate the potential of LLM-driven reasoning to manage conflicts in complex and dynamic RAN systems.**

I. Introduction

The ORAN paradigm promotes a disaggregated, virtualized, and multi-vendor mobile network architecture, enabling flexibility and innovation through standardized open interfaces. Central to this design are the RAN Intelligent Controllers (RICs)—non-Real-Time (non-RT) and near-Real-Time (near-RT)—which host independently developed software applications (rApps and xApps) for network optimization. While openness drives innovation, it can also introduce unintended interactions or conflicts among applications that share a common base of network resources.

Recent efforts in conflict detection primarily rely on statistical methods. For instance, the state-of-the-art PACIFISTA method [1] uses empirical cumulative distribution functions (ECDFs) to quantify conflict severity and decides whether to allow the coexistence of applications before deployment. Another approach [2] uses statistical correlation to predict KPI impacts from configuration changes, preemptively deciding whether to apply the configuration changes. Despite these advances, several fundamental limitations persist. First, existing methods lack fine-grained identification of conflicting parameters, forcing coarse-grained decisions where entire change sets are accepted or rejected wholesale. Second, their reliance on static historical data makes them ineffective in unseen environments.

Fig. 1a elucidates such limitations by experimenting with PACIFISTA [1]. Two xApps control physical resource block (PRB) allocation with different objectives: xApp A maximizes eMBB throughput, while xApp B minimizes URLLC delay. We compare the ECDFs (y-axis) of eMBB PRB allocations (x-axis) across two environments with *different cell loads and traffic patterns*. In one environment (left), their ECDFs are similar, indicating compatible behavior. In the other (right),



(a) ECDF of different xApps (b) Actual vs Predicted Tput

Fig. 1: Conflict detection methods do not hold in unseen environments.

they diverge significantly, showing that xApp coexistence breaks under environmental changes, as also noted in [1]. PACIFISTA addresses this by collecting conditioned statistical profiles for each environment, which allows it to predict the impact of resource usage with potential conflicts. Fig. 1b shows PACIFISTA's prediction of throughput under network environment dynamics. We trained a model using data from different environments with varying parameters (e.g., Tx-Power, traffic load). The left and right subfigures show the prediction results by including and excluding one particular (unseen) environment, respectively. We observe poor generalization to unseen conditions, which also hampers PACIFISTA's ability of conflict detection under environmental changes. These experiments motivate the need for a framework that can adapt to new environments without requiring retraining or collecting statistical profiles for each xApp.

In response, we introduce Open RAN Conflict Agents (ORCA), the first framework to employ LLM agents for conflict detection. To enable fine-grained conflict identification, ORCA estimates ***feature importance***, defined as the sensitivity of a KPI to variations in individual control parameters or network metrics. ORCA computes this using SHapley Additive exPlanations (SHAP) [3], a well established method for estimating feature importance. However, directly applying SHAP to the full set of ORAN parameters is impractical due to high dimensionality of the parameter space. ORCA addresses this via a *two-stage pre-selector*: an autoencoder removes irrelevant features, and an LLM agent recovers those critical for KPI prediction. This pre-selector focuses SHAP on the most relevant subset for efficient and accurate feature attribution.

The second challenge, adapting to dynamic and evolving network environments, is considerably more complex. While one might rerun SHAP analysis in a new environment, this would require a large and diverse volume of log data to produce stable and meaningful feature attributions, which can take hours to days to collect and can easily expire as network condition changes. ORCA addresses this limita-

tion by employing generative LLM agents to infer feature importance in unseen environments without requiring such extensive data. Given SHAP values (feature importance) from a known environment, ORCA guides the LLM agents to reason about how feature importance might shift under new conditions. Unfortunately, the LLM often produces heuristic rather than quantitatively grounded outputs. ORCA overcomes this limitation through a β-*Reasoning Cascade*, a structured process that supports symbolic reasoning for deriving robust and interpretable feature importance.

Another challenge is the risk of hallucination, which may result from missing domain knowledge or, even when the knowledge exists, from the LLM's internal reasoning errors or faulty synthesis. To mitigate hallucinations and maintain stability, we propose Segmented Active RAG (SA-RAG)—a dynamic retrieval method that adapts queries as reasoning evolves and targets domain-specific knowledge sources such as 3GPP specifications, keeping the LLM grounded in context-relevant information throughout the process.

We carefully constructed 32 categories of network environments to capture a broad and representative range of variations across key factors such as control parameters, traffic patterns, cell load, channel conditions, and UE mobility. While not exhaustive, this set reflects meaningful diversity across realistic operational conditions, providing strong coverage for evaluating the generalization capability of our framework. We compared ORCA with two baselines: Extrapolation, and Pure LLM, by using SHAP's results as ground-truth. We found that ORCA achieved the most accurate estimates, reducing cosine error by 60.7% over Extrapolation and NRMSE by 65.1% over Pure LLM. We also tested scenarios involving resource conflicts, where we compared ORCA with PACIFISTA [1] and a prediction-based method inspired by [2]. Our experiments indicate that ORCA enables fine-grained, context-aware control by selectively rejecting actions that threaten critical KPIs.

In summary, the main contributions of this paper are:

- We propose ORCA, a fine-grained ORAN conflict detection framework designed for unseen and dynamically changing network environments.
- We design a two-stage pre-selector scheme, which combines dimension reduction techniques with LLM agents for generalizable conflict analysis. In addition, we present the β-Reasoning Cascade and SA-RAG, two novel mechanisms that empower LLM agents to perform accurate multi-step reasoning, by effectively retrieving ORAN specific domain knowledge.
- We verify the effectiveness of ORCA across diverse network environments, in comparison with state-of-the-art baselines.

II. Background

A. Conflicts in ORAN App Configuration

Consider an ORAN deployment consisting of a set L xApps: $\mathcal{X} = \{x_1, x_2, \ldots, x_L\}$, M control parameters: $\mathcal{P} = \{p_1, p_2, \ldots, p_M\}$, and N KPIs: $\mathcal{K} = \{k_1, k_2, \ldots, k_N\}$. Different xApps often target diverse objectives (i.e., different KPIs) and control different sets of parameters, but fundamentally they are sharing a common pool of network resources. So their operations can unintentionally interfere with each other. The ORAN Alliance specification [4] classifies such *conflicts* among xApps into three categories.

(i) Direct conflict: Occurs when two or more xApps control the same parameters or compete for the same RAN resources.
(ii) Indirect conflict: Occurs when multiple xApps control different parameters to optimize the same metric.
(iii) Implicit conflict: Occurs when multiple xApps control different parameters to optimize different metrics, yet their actions inadvertently interfere with each other.

ORCA mitigates all conflict types by detecting potential parameter conflicts before xApps apply them.

B. SHAP Analysis for Feature Importance

Understanding the contribution of parameters to KPIs is critical for fine-grained conflict detection. The most rigorous way to quantify this contribution is to perturb the target parameter and measure its impact on the KPI across all possible combinations of other parameters. This forms the core principle of the original Shapley value framework [5]. Let $\mathcal{F}$ denote the selected feature set (parameter set). The feature i's importance ϕ_i is formally defined as:

$$\phi_i = \sum_{S \subseteq \mathcal{F} \setminus \{i\}} \frac{|S|!\,(|\mathcal{F}| - |S| - 1)!}{|\mathcal{F}|!} \left[f(S \cup \{i\}) - f(S) \right], \quad (1)$$

where f denotes the expected output of the surrogate model. Here, feature importance is interpreted as a weighted average of the marginal contributions of feature i across all possible subsets $S \subseteq \mathcal{F}$. However, computing exact Shapley values is infeasible in practice for two reasons. First, without a trained model, perturbing parameters requires applying each change to a real system or simulator to observe its effect on the KPI, which is extremely costly. Second, the number of possible parameter subsets grow exponentially with the number of parameters, making exhaustive evaluation impractical.

SHAP [3] addresses both challenges by using a trained model to efficiently approximate Shapley values and by applying sampling-based methods like Kernel SHAP to reduce computational complexity. We apply SHAP to quantify how changes in parameters affect KPIs. In line with SHAP terminology, we treat parameters as "features", and refer to the degree of contribution as "feature importance". This enables fine-grained estimation of individual parameter's impact, critical for identifying potential conflicts between xApps.

However, SHAP validity fundamentally depends on model accuracy. SHAP values reliably reflect feature importance within the environment the model was trained on, where the environment refers to external operating conditions such as traffic load, channel condition, and mobility status, but may become unreliable under unseen conditions. Consequently, when network environments change, reusing SHAP values from prior conditions can lead to incorrect feature impact assessments. Maintaining reliable conflict detection in dynamic environments requires generating or inferring updated SHAP values that reflect the current context.

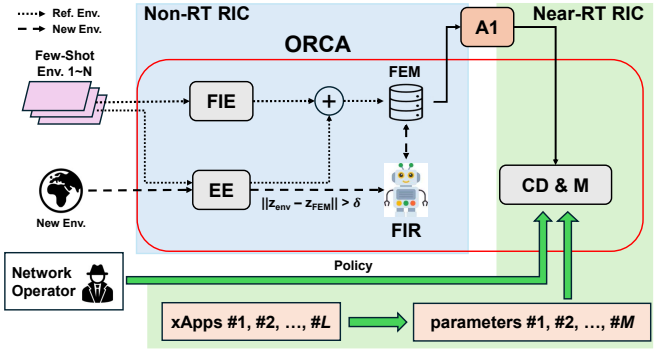


Fig. 2: Overall architecture of ORCA, consisting of FIE, EE, FIR, and CD&M.

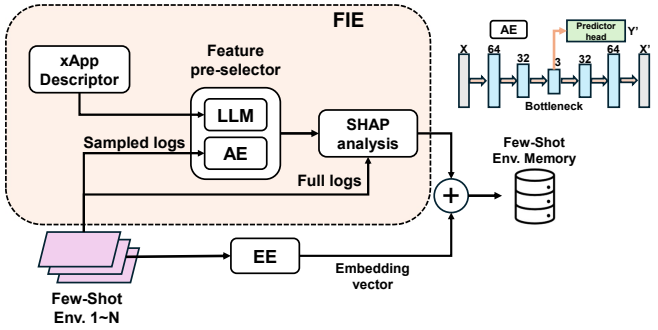


Fig. 3: Detailed structure of the FIE.

III. System Design

A. ORCA Architecture

As illustrated in Fig. 2, ORCA consists of four key modules: the Environment Encoder (EE), the Feature Importance Extractor (FIE), the Feature Importance Reasoner (FIR), and the Conflict Detection and Mitigation (CD&M) module. EE processes raw contextual features and transforms them into a compact environment embedding. FIE applies SHAP analysis to extract each feature's importance, while FIR uses LLM agents to derive updated feature importance. Finally, the CD&M module detects and mitigates conflicts based on the resulting feature importance.

ORCA first ingests few-shot ORAN logs from known environments ("shot" refers to the number of environments), which we refer to as *reference environments*. The logs include control parameters set by xApps or network operator (e.g., TxPower), metrics (e.g., SNR), and KPIs (e.g., throughput). Since conflict detection is performed by the network operator, we reasonably assume that such logs are readily available.

EE and FIE are applied to each reference environment. EE generates a compact embedding of the environment context, while FIE performs SHAP analysis to compute the importance of each control parameter. Both the embeddings and the corresponding feature importance vectors are stored in the Few-Shot Environment Memory (FEM) and passed to the CD&M module via the A1 interface [6]. To detect environmental shifts, EE is executed periodically at a fixed interval T, producing an embedding vector $\mathbf{z}_{\text{env}}$ for the current environment. This embedding is compared against embeddings $\mathbf{z}_{\text{FEM}}$ stored in the FEM. If $\|\mathbf{z}_{\text{env}} - \mathbf{z}_{\text{FEM}}\| > \delta$, where δ is a predefined threshold, ORCA infers that a previously unseen environment has emerged.

Upon detecting a new environment, FIR is invoked to derive updated SHAP values that reflect the new network conditions. These updated feature importance vectors are then stored in the FEM and passed to the CD&M module via the A1 interface. The CD&M module continuously monitors control parameter change requests from xApps, predicts their impact on KPIs using the updated SHAP values, and decides whether to approve or reject each request.

The overall system design reflects practical deployment considerations within ORAN. EE, FIE, and FIR are designed to reside in the non-RT RIC, as they rely on historical log data and do not require low-latency execution. In contrast, the CD&M module, which performs live conflict mitigation, is deployed in the near-RT RIC to satisfy latency constraints. This division of responsibilities aligns well with execution times. The FIR takes approximately 33 seconds to compute new feature importance values, which is acceptable given that environmental changes in cellular networks typically occur over timescales of hours. Meanwhile, the CD&M module performs only lightweight arithmetic over a small number of features, enabling decision times below one millisecond, meeting near-RT latency needs.

B. Environment Encoder (EE)

The network environment is dynamic and evolves over time. To capture this variability, we employ an EE that maps the current environment into a compact embedding summarizing key contextual attributes. The encoder processes features such as cell-level traffic load, channel condition, slice configuration, and mobility status to produce a representation of the network state. These features are typically accessible to the network operator through standard O1, with cell-level metrics changing slowly enough that reporting intervals of tens of seconds provide sufficient context for environment detection. This embedding serves as contextual grounding for downstream reasoning tasks such as β-propagation, enabling generalization to previously unseen environments.

C. Feature Importance Extractor (FIE)

Accurate feature-importance extraction is vital for conflict management, as it shows how each parameter influences KPIs. FIE employs SHAP [3] to extract feature importance in the reference environments. However, applying SHAP directly to ORAN data, which often contain hundreds of features, faces two major challenges. First, SHAP's computational complexity grows exponentially with input size: irrelevant features force the explainer to evaluate many unnecessary coalitions, severely slowing analysis. Second, training a reliable surrogate model on high-dimensional data with numerous irrelevant inputs risks overfitting or demands substantially more samples; a poorly performing model yields misleading SHAP values.

To address these challenges, we introduce a feature pre-selector combining a supervised autoencoder (AE) and an

LLM agent. The AE is trained on a sampled subset of logs (e.g., 20% of the ORAN logs) to rapidly filter out irrelevant features prior to SHAP analysis. We empirically observed that unsupervised training leads to inaccurate feature importance ordering, as the latent representation is optimized solely for input reconstruction rather than for preserving KPI-relevant feature relationships. Therefore, we augment the AE with a KPI-prediction head to guide the latent representation toward KPI-relevant feature relationships. As illustrated in Fig. 3, the encoder consists of three fully connected layers and a decoder mirrors this structure in reverse to reconstruct the original input vector. In parallel, a predictor head estimates the KPI directly from the bottleneck. Training minimizes a joint loss:

$$L(\theta) = \lambda \left\|\hat{\mathbf{x}} - \mathbf{x}\right\|_2^2 + (1-\lambda)\left(\hat{y} - y\right)^2 + \alpha \sum_{i,j} \left|W_{ij}^{(7)}\right|,$$

where $\mathbf{x} \in \mathbb{R}^D$ is the input feature vector and $\hat{\mathbf{x}}$ is its reconstruction. The scalar y is the ground truth KPI (e.g., throughput), and $\hat{y}$ is the predicted KPI. The matrix $W^{(7)}$ denotes the weight matrix of the first linear layer in the predictor head. The term $\lambda = 0.3$ balances the reconstruction loss and KPI prediction loss, while $\alpha = 10^{-4}$ controls the sparsity regularization. The total trainable parameters of the model are denoted by θ. The reconstruction loss encourages the model to preserve features necessary for reconstructing the overall input structure, while the KPI prediction loss focuses the model on features that directly influence the target KPI.

During training, features that contribute to neither reconstruction nor KPI prediction objectives are naturally suppressed by the AE. After training, we compute the gradient of the loss with respect to each input feature to quantify their influence. Features whose gradient magnitudes fall below a relative threshold (e.g., 10% of the top-ranked feature) are discarded. The final output of this stage is a reduced set of selected features, prioritized for their joint contribution to both input reconstruction and KPI prediction. However, this process may inadvertently eliminate infrequent yet important features.

To address this limitation, we incorporate an LLM agent guided by the developer-provided xApp descriptors, which specify the controllable parameters and the targeted KPIs. The LLM uses these descriptors as structured constraints and applies domain knowledge to identify parameters that have a documented or plausible causal relationship with the target KPIs, even if they are underrepresented in the sampled data. It cross-references the xApp's targets with the full parameter set to recover important parameters that may have been excluded earlier, ensuring that critical features are not overlooked. This ensures that all relevant parameter–KPI relationships are retained for SHAP analysis.

After pre-selection, we train a surrogate model using the reduced feature set and perform SHAP analysis to compute each feature's contribution to the KPI. Since exact computation is intractable (see §II-B), we adopt approximation methods such as Kernel SHAP [3] using standard SHAP libraries. SHAP is applied to both control parameters and network metrics. The resulting values, along with the environment embedding, are stored in the FEM.

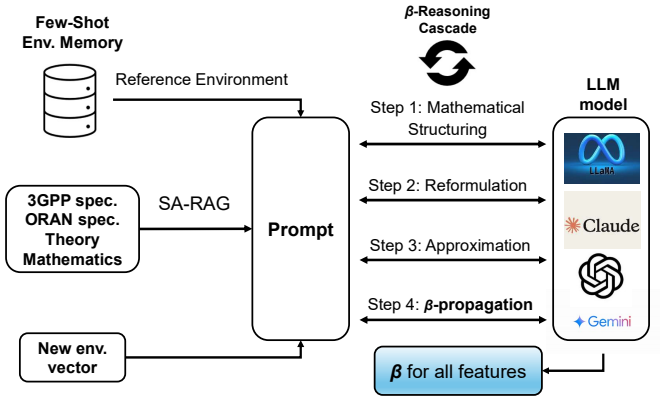


Fig. 4: FIR module infers new importance using β-Reasoning Cascade and external knowledge retrieved via SA-RAG.

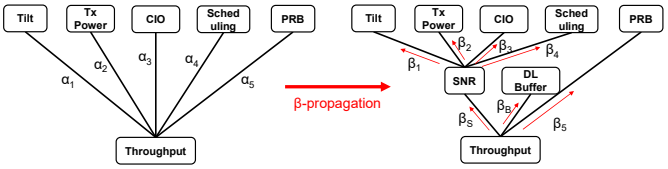


Fig. 5: Feature importance is derived by SHAP analysis (left), while new feature importance is obtained via β-propagation (right). β values are propagated through intermediate metrics.

D. Feature Importance Reasoner (FIR)

Although we have feature importance of reference environment, deriving SHAP values for a new, unseen environment remains challenging. Conventional methods like extrapolation, regression, or ML prediction are inadequate: extrapolation fails under non-linear dynamics, and regression or ML requires extensive SHAP supervision across diverse conditions. A natural alternative is principled mathematical reasoning, as a domain expert might do.

Suppose the target KPI is throughput and the features are `TxPower`, `PRB`, `DL_buffer`, and `antenna tilt`. Since throughput cannot be directly expressed as a function of these parameters, experts often use analytic formula such as Shannon's capacity and recursively reason over lower-level features. Feature importance is then estimated using partial derivatives to assess KPI sensitivity. When closed-form models are unavailable or too complex, experts rely on approximations based on feature distributions. However, in systems like ORAN, this process becomes intractable due to high dimensionality and implicit dependencies. Even if an approximation is available, validating its applicability under distributional shift remains a major challenge. We address this by delegating the structured expert reasoning process to an LLM agent, implemented through the β-Reasoning Cascade.

*1) **β-Reasoning Cascade**:* One might expect a powerful LLM to derive feature importance through symbolic reasoning by manipulating mathematical relationships. However, without structured guidance, LLMs often fail to perform this reliably. To address this, FIR uses a structured reasoning process, the ***β-Reasoning Cascade***. Rather than following a rigid pipeline, FIR leverages the LLM agent's reasoning capabilities

to dynamically construct, transform, and validate logic at each stage. Unlike human experts, who struggle to repeat symbolic analysis across many environments, the LLM agent efficiently scales this reasoning by combining symbolic manipulation with domain-informed generalization. As shown in Fig. 4, the reasoning framework is composed of 4 steps. At each step, the LLM agent leverages its reasoning based on prompt, domain specific and pretrained knowledge. The necessary domain specific knowledge is retrieved via SA-RAG (see §III-D2).

Step 1: Mathematical Structuring. Each KPI's definition is provided to the LLM agent via prompt. For example:

- *Throughput*: total amount of user data successfully transmitted to all UEs in a cell per second (Mbps)
- *RLFs*: events where SNR remains below a defined threshold γ for a specified time duration T

Together with the given information and knowledge, the LLM agent formulates a symbolic expression of the KPI as a function of abstract system variables, such as Throughput $=$ $\min$(System Capacity, Traffic Demand). In this step, the primary source of domain knowledge is often communication theory, as observed in our experiments. However, for certain stochastic KPIs (e.g., RLFs), LLM agent determines that no deterministic symbolic form exists and therefore skips Step 1, proceeding directly to Step 2.

Step 2: Reformulation. The symbolic expression is then rewritten in terms of measurable parameters, including network metrics and control parameters. For example, system capacity may be expressed as a function of `Avg_SNR` and `PRB_num`, while traffic demand may be approximated using `DL_buffer`. For KPIs where Step 1 is skipped, the LLM proceeds directly to Step 2, identifying dominant measurable parameters that serve as the basis for probabilistic modeling. In this step, the primary source of domain knowledge is typically 3GPP and ORAN specifications.

Step 3: Approximation. Although Step 2 yields a reformulated expression in terms of measurable parameters, the resulting equation is often not in closed form. This is particularly common for network KPIs, which are frequently expressed as probabilities or expectations. Moreover, partial derivatives can be applied to the expression to assess the sensitivity of the KPI to different control parameters. This often results in intractable or non-analytic terms. In such cases, mathematical approximation techniques are employed to convert the expression into a closed form. In this step, the primary source of domain knowledge is mathematics.

Step 4: β-Propagation. We use the importance shift ratio, denoted by β, instead of directly deriving the absolute SHAP values denoted by ϕ for the new environment. β describes how the importance of a feature changes relative to a reference environment when evaluated in a new environment. This design choice is motivated by the fact that approximations are applied at multiple stages of the reasoning process—particularly in step 2 and 3, where the KPI is reformulated and simplified into closed-form expressions. If we attempt to compute the absolute SHAP values directly in the new environment, the accumulated approximation error can lead to inaccuracies. In contrast, computing the importance shift ratio β helps mitigate this issue: since both the numerator and denominator undergo the same symbolic transformations, approximation errors tend to "cancel out" in the ratio. Furthermore, to enable robust reasoning within the LLM agent, we use normalized SHAP values α instead of raw values ϕ, as ϕ is often proportional to the KPI. While SHAP is not mathematically equivalent to a partial derivative, it can be conceptually approximated, as both quantify the local sensitivity of the model output with respect to a given input feature. In particular, the magnitude of the partial derivative serves as a reasonable proxy for the normalized SHAP value:

$$\alpha_i^{\text{ref}} = \frac{|\phi_i^{\text{ref}}|}{\sum_{j\in\mathcal{F}} |\phi_j^{\text{ref}}|} \approx \frac{\left|\frac{\partial\,\text{KPI}}{\partial i}\right|_{\text{ref}}}{\sum_{j\in\mathcal{F}} \left|\frac{\partial\,\text{KPI}}{\partial j}\right|_{\text{ref}}} \tag{2}$$

$$\beta_i = \frac{\alpha_i^{\text{new}}}{\alpha_i^{\text{ref}}} \approx \frac{\left|\frac{\partial\,\text{KPI}}{\partial i}\right|_{\text{new}} \Big/ \sum_{j\in\mathcal{F}} \left|\frac{\partial\,\text{KPI}}{\partial j}\right|_{\text{new}}}{\left|\frac{\partial\,\text{KPI}}{\partial i}\right|_{\text{ref}} \Big/ \sum_{j\in\mathcal{F}} \left|\frac{\partial\,\text{KPI}}{\partial j}\right|_{\text{ref}}} \tag{3}$$

In the example of throughput, β for features such as PRB can be directly derived using Eq. (3), since PRB appears explicitly in the throughput formulation. However, other features such as `TxPower`, or `antenna tilt`, influence throughput indirectly through intermediate variables like SNR. In such cases, their impact is embedded within the shift ratio of the intermediate metric. For instance, $\beta_{TxPower}$ is implicitly captured by β_{SNR}, and must be resolved recursively to recover the original feature's contribution. As illustrated in Fig. 5, this recursive tracing of influence from intermediate variables back to original features is referred to as β-propagation. The actual values of β are conditioned on the current environment, and are computed based on the low-dimensional embedding produced by the EE. We emphasize again that, although the reasoning process follows a structured four-step framework, each step is autonomously performed by the LLM agent, not through fixed logic. The degree of reasoning varies: Steps 1 requires light symbolic mapping, whereas Step 2, Step 3, and 4 demand deeper inference, including approximation and distribution-aware validation.

Once the β_i values for all features are reasoned by FIR, we compute the new normalized feature importance by applying the shift ratio, $\alpha_i^{\text{new}} = \beta_i \cdot \alpha_i^{\text{ref}}$. We then renormalize to ensure that the normalized importance values sum to one: $\widetilde{\alpha}_i^{\text{new}} = \frac{\alpha_i^{\text{new}}}{\sum_{j\in\mathcal{F}} \alpha_j^{\text{new}}}$. Finally, we rescale the renormalized importance values to obtain the absolute SHAP values by multiplying with the KPI value of the new environment:

$$|\phi_i^{\text{new}}| = \widetilde{\alpha}_i^{\text{new}} \cdot \left(\frac{\text{KPI}^{\text{new}}}{\text{KPI}^{\text{ref}}}\right) \cdot \sum_j |\phi_j^{\text{ref}}| \tag{4}$$

Note on KPI applicability: reasoning pipeline is particularly effective for KPIs like throughput and delay, where domain knowledge aligns well with observable relationships in the data, enabling the model to reason effectively about control parameter impact. For more stochastic or higher-layer KPIs (e.g.,

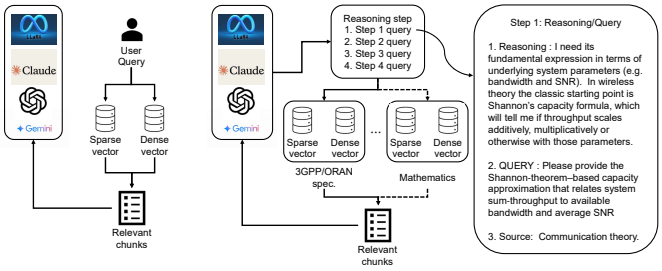


Fig. 6: Conventional hybrid RAG (left) vs SA-RAG (right).

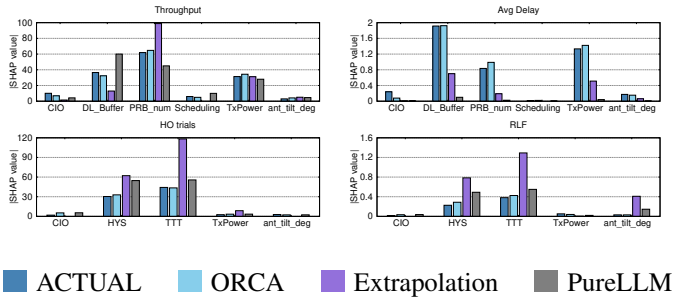


Fig. 7: Comparison of derived feature importance against ground-truth (ACTUAL) values. Closer alignment indicates higher estimation accuracy.

reliability or user satisfaction), additional domain modeling or assumptions may be required to capture indirect dependencies.

*2) **Knowledge-Augmented Reasoning**:* Another challenge in reasoning feature importance is hallucination, where an LLM generates incorrect or unfounded outputs. To mitigate such risks, we provide domain knowledge through a hybrid RAG framework combining sparse and dense retrieval. Sparse retrieval locates exact keyword matches while dense retrieval captures semantic similarity and retrieves relevant information even when the query and document use different terminologies or phrasing. By combining both, hybrid RAG ensures comprehensive and contextually accurate knowledge grounding for the LLM. However, conventional hybrid RAG methods are passive, meaning that retrieval queries are either manually written or generated from the initial prompt without adapting to intermediate reasoning needs or environmental context. As a result, they often retrieve loosely relevant information, making them insufficient for multi-step reasoning tasks.

To address this limitation, we propose Segmented Active RAG (SA-RAG), a reasoning-aware RAG framework in which the LLM dynamically generates retrieval queries during the reasoning process. As shown in Fig. 6, the LLM formulates queries at each stage based on the current reasoning objective. For instance, it may first retrieve Shannon's capacity formula to relate throughput, bandwidth, and SNR, then refine queries to focus on parameter behaviors or constraints. While the LLM may have learned such knowledge during pretraining, it can produce incorrect or inconsistent outputs due to reasoning errors. To mitigate this, our framework explicitly instructs the LLM to retrieve external domain knowledge via SA-RAG, enabling it to combine pretrained and retrieved knowledge for more accurate decisions.

During the β-Reasoning Cascade, relevant knowledge spans 3GPP and ORAN specifications, mathematics, communication theory, and system-level information. Mixing these domains in a single retrieval pool often leads to irrelevant results. Thus, SA-RAG segments knowledge sources and routes each query to the most appropriate domain, improving retrieval precision.

E. Conflict Detection and Mitigation (CD&M)

As mentioned earlier, a major advantage of extracting feature importance is that it enables network operators to make policy-driven decisions by estimating the impact of feature changes on KPIs. This process begins with quantifying the expected KPI change resulting from a proposed parameter adjustment. We begin by interpreting the SHAP value ϕ_i as the expected absolute change in the KPI due to perturbing feature i: $\phi_i \approx \mathbb{E}\left[|\Delta\text{KPI}_i|\right]$. Assuming local linearity of the KPI with respect to feature i, $\Delta\text{KPI}_i \approx \frac{\partial\,\text{KPI}}{\partial i} \cdot (i - i_{\text{baseline}})$, where i_{baseline} denotes the baseline value of feature i in the reference environment. It represents the operating point at which the SHAP value ϕ_i was originally computed. Taking the expectation of the absolute value, we obtain $\phi_i \approx \left|\frac{\partial\,\text{KPI}}{\partial i}\right| \cdot \epsilon_i$ where $\epsilon_i := \mathbb{E}[|i - i_{\text{baseline}}|]$ is the expected magnitude of deviation from the average value of feature i in the reference environment. We use ϵ_i from the reference environment since the new SHAP value is derived from the reference distribution of feature i. Now, given a specific proposed change Δi in feature i, the expected impact on the KPI is:

$$\Delta\text{KPI}_i \approx \frac{\phi_i}{\epsilon_i} \cdot \Delta i \tag{5}$$

When parameter changes are requested by xApps, the impact on the KPI can be estimated using Eq. (5). A conflict is declared if $\Delta KPI > \delta$, where the δ is a predefined threshold by the network operator. Depending on the predefined network operator policy and prioritization rules, the proposed parameter change is either accepted or rejected accordingly.

IV. Evaluation

A. Simulation Setup

ORCA is powered by an LLM agent based on o4-mini [7], serving as the reasoning engine for conflict detection. We evaluate ORCA using a MATLAB-based 5G simulator with the 5G Toolbox. While this is a simple simulator, it provides sufficient flexibility to model key aspects of RAN slicing, including per-slice PRB allocation, traffic generation, and KPI logging. The setup includes two gNBs and up to 100 UEs, operating over 100 MHz (273 PRBs) partitioned between eMBB and URLLC slices. ORCA treats cross-cell influences as part of the environment; for example, if an xApp controls only cell A, we monitor the KPIs of cell A, whereas if it controls both A and B, we monitor the combined KPIs of both cells. Although our setup includes only two gNBs, this design naturally extends to scenarios with a larger number of cells. Moreover, we define only one eMBB slice and one URLLC slice, but each slice is not limited to just a single flow. Instead, it can represent a group of multiple traffic flows that have similar service requirements. We measure four KPIs: eMBB downlink throughput, URLLC average delay, number of handovers (HOs), and radio link failures (RLFs). UE mobility includes back-and-forth movement, random direction

TABLE I: Overview of Applications and Controlled Parameters

Application	Objective	Control Time Scale (slots)	Controlled Parameters
Throughput maximization (TM)	Maximize throughput	200	PRBs, TxPower
Energy Saving (ES)	Reduce energy consumption	200	TxPower
Delay Minimization (DM)	Minimize end-to-end delay	200	PRBs, scheduling
Handover Management (HM)	Reduce number of handovers	500	TTT, Hysteresis
Mobility Robustness Optimization (MRO)	Reduce RLFs (radio link failures)	500	Antenna Tilt, CIO

changes, and stationary behavior. URLLC traffic follows a Poisson process (1–5 packets/s), while eMBB users generate continuous traffic to simulate high throughput.

We applied Kernel SHAP [3] on models including XGBoost [8] and MLP to derive ground truth feature importance. As benchmarks, we used extrapolation and a pure LLM agent. Regression and ML-based predictions require large labeled datasets across diverse environments, which contradicts our goal of generalization to unseen conditions.

We quantify accuracy using cosine error and normalized root mean square error (NRMSE). Cosine error captures the directional similarity of predicted and ground-truth importance vectors, while NRMSE measures the magnitude difference. Together, they evaluate both relative and absolute accuracy. We also evaluate ORCA's conflict management with various xApps as indicated in Table I. We benchmark against PACIFISTA [1] and a prediction-based method, inspired by [2].

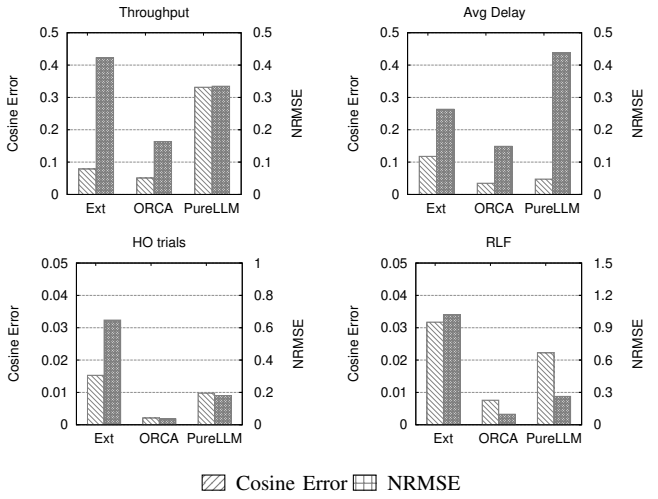


Fig. 8: Comparison of cosine error and NRMSE. "Ext" denotes the extrapolation baseline.

B. Results

Our evaluation consists of two parts: (i) the accuracy of feature importance, measured by NRMSE and cosine error against ground truth, and (ii) conflict detection and mitigation, evaluated through KPI performance. The latter inherently involves trade-offs across KPIs, which vary depending on the network operator's priorities. *Therefore, rather than merely presenting statistical KPIs, we showcase why ORCA makes certain decisions under conflicting control requests.* Since improving all KPIs simultaneously is infeasible, our focus is on explainable, priority-aware decision-making.

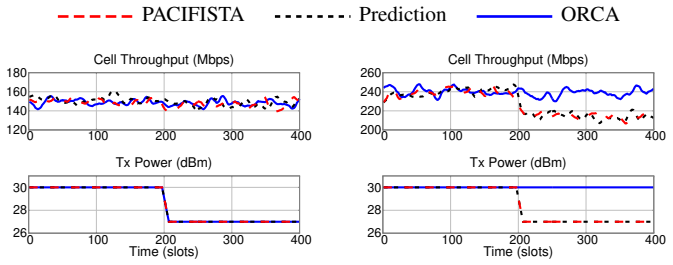
1) ***Accuracy of feature importance****:* We create an additional set of 32 network environments by systematically perturbing control parameters and environmental factors. From a single reference environment, both ORCA and the pure LLM agent are provided with the true feature importance in a one-shot manner. In contrast, the extrapolation-based method requires at least two known environments to function, so we provide it with additional ground-truth importance values accordingly. Fig. 7 presents inferred SHAP values for each parameter in one unseen environment. As can be visually confirmed, ORCA most closely matches the actual feature importance. For all 32 environments, each method generates its own feature importance estimates, which are compared against the ground truth. As shown in Fig. 8 and Table II, ORCA achieves the lowest error across all KPIs, outperforming the best baseline by 60.7% in cosine error and 65.1% in NRMSE.

2) ***Conflict Detection and Mitigation****:* We first consider a conflict scenario between the ES and TM xApps. The ES xApp requests a reduction in TxPower to conserve energy. In Fig. 9a, all methods accept the change, as the expected impact on throughput is minimal in this environment. From our experiments, this situation tends to occur when eMBB traffic demand is relatively low or the cell is lightly loaded, resulting in sufficient PRB allocation to the eMBB slice. In such cases, a moderate TxPower reduction does not significantly impact throughput. However, as shown in Fig. 9b, in a new environment with higher traffic load or near-full PRB utilization, ORCA correctly rejects the TxPower reduction, anticipating substantial throughput degradation under these conditions.

We then consider a conflict scenario between the HM and MRO functionalities. PACIFISTA does not allow the coexistence of conflicting xApps in this environment; therefore, only prediction-based methods are considered for comparison. In Fig. 10a, both methods reject the parameter change, as increasing the hysteresis is predicted to raise the number of RLFs. However, in a different environment (Fig. 10b), while the prediction-based method still rejects the change, ORCA accepts it. As a result, ORCA reduces the number of ping-pong handovers while maintaining the RLF rate. We observed this behavior particularly when UE mobility was reduced, such that RLFs were less sensitive to hysteresis adjustments. In our simulation, we assume the ORAN system does not directly observe UE mobility, but SNR measurements from sounding reference signals (SRS) are available. FIR leverages the variation of SNR to infer the feature importance of hysteresis under such conditions.

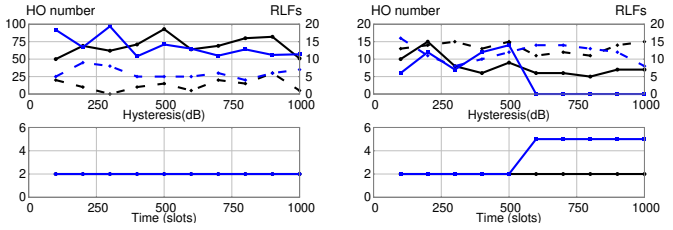
TABLE II: Average cosine error and NRMSE across all 32 unseen environments and 4 KPIs.

	Extrapolation	Pure LLM	ORCA(ours)
Cosine error	0.061	0.103	**0.024** (↓60.7% vs EXT)
NRMSE	0.589	0.300	**0.105** (↓65.1% vs LLM)



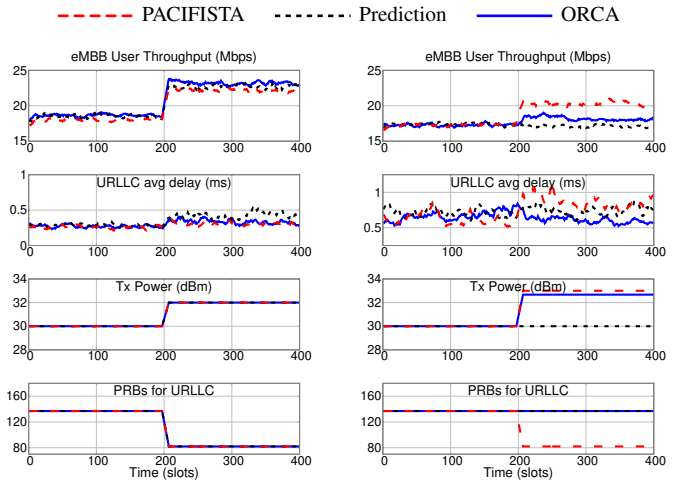
(a) (b)

Fig. 9: Conflict scenario between ES and TM.



(a) (b)

Fig. 10: Conflict scenario between HM and MRO. In the top subplots, solid lines show the number of handovers and dashed lines show the number of RLFs.



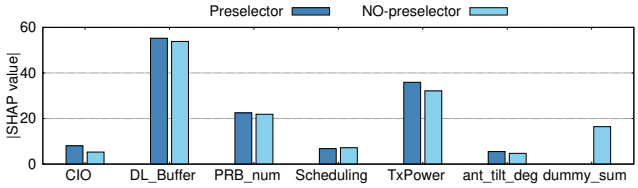
(a) (b)

Fig. 11: Conflict scenario between TM and DM.

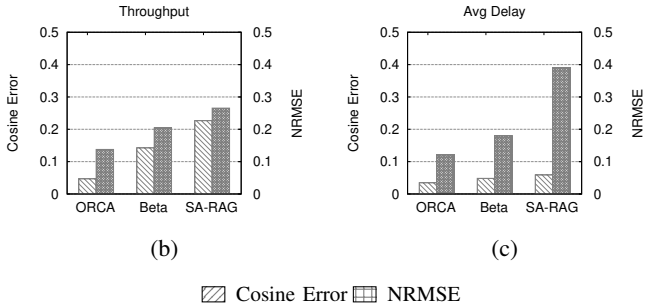
Finally, we examine a conflict scenario between the TM and DM xApps. The TM xApp requests an increase in both TxPower and PRBs to boost throughput. In Fig. 11a, all methods accept the change, as the expected impact on delay is relatively minor. We observed that when URLLC traffic is low and PRBs are sufficiently provisioned to URLLC, the DM xApp also tends to accept additional PRBs for eMBB without causing conflict. However, in a new environment (Fig. 11b), while PACIFISTA accepts the change, both the prediction-based method and ORCA detect potential conflicts. The prediction-based method rejects all requested changes, whereas ORCA makes a finer decision: it rejects the PRB increase because allocating more PRBs to eMBB reduces availability for URLLC, but accepts the TxPower increase.

*3) **Ablation study**:*

Impact of the feature preselector To evaluate the impact



(a) Ablation study on throughput in one example environment, comparing results with and without the preselector.



(b) (c)

Fig. 12: (b) and (c) show the accuracy of feature importance estimation for throughput and average delay, respectively. ORCA (full system) is compared against Beta (without SA-RAG) and SA-RAG (without the β-Reasoning Cascade).

of feature preselector, we compared two scenarios: one using only real control and metric features ($n = 8$), and another with 64 additional dummy features ($n = 72$). Since SHAP becomes infeasible at high dimensionality, so we sampled 2,000 subsets for both cases.[1] As shown in Fig. 12a, dummy features absorbed 11.5% of total SHAP importance despite having no causal relevance. While individual SHAP values were small, their cumulative effect was non-negligible We note that the 11.5% dummy feature share reflects the variance inherent in the sampling-based Kernel SHAP method at the chosen evaluation budget, and is expected to decrease further with larger sample sizes. In contrast, the preselector removed dummy features, resulting in more stable and concentrated importance on key variables, confirming reduced noise and preventing SHAP dilution across irrelevant features. We then compared the SHAP computation time. Despite using the same number of subset evaluations, SHAP analysis took 576s for $n = 72$, compared to 28s for $n = 8$, both using only 800 log samples. While the absolute computation time depends on hardware, the relative increase of approximately 20 times remains consistent across hardware. We attribute this to the larger model size and the increased masking overhead associated with the larger feature set, both of which substantially raise the per-sample computation cost during SHAP analysis.

We also evaluated whether important features can be reinstated when they are discarded by AE. For RLFs, `ant_tilt_deg` was removed by the AE as part of the dimensionality reduction process. However, LLM agent rescues it. Although `ant_tilt_deg` exhibited low importance with

[1]For $n = 8$, we intentionally oversampled beyond the full subset space of $2^8 = 256$ to match the experimental conditions of the large-n case. For $n = 72$, where the full subset space (2^{72}) is computationally infeasible, we heavily undersampled to 2,000 evaluations to enable direct runtime comparison.

β-Reasoning Cascade (part of step 3)

1. Probabilistic formulation
Let $C = \min(A, B)$. A perturbation in A affects C only if $A < B$; similarly for B. The importance of A is defined as

$$X \coloneqq \mathbb{E}\big[|\partial C/\partial A|\big] = P(A < B)$$

$$Y \coloneqq \mathbb{E}\big[|\partial C/\partial B|\big] = P(B < A)$$

Letting $D = A - B$,

$$P(A < B) = F_D(0), \quad P(B < A) \quad = 1 - F_D(0),$$

2. Closed-form ratio approximation
To simplify, we approximate win probabilities using the means $a = \mathbb{E}[A]$, $b = \mathbb{E}[B]$. Define

$$\Delta\gamma \coloneqq \frac{a'}{b'} \Big/ \frac{a}{b}.$$

Then estimate

$$\beta_A \coloneqq \frac{X'}{X} \approx \frac{1}{\Delta\gamma}, \quad \beta_B \coloneqq \frac{Y'}{Y} \approx \Delta\gamma.$$

Thus, if A grows relative to B, B's importance increases by that factor, and A's decreases reciprocally.

Fig. 13: Example from the β-Reasoning Cascade (step 3)

SHAP analysis in one environment, ant_tilt_deg represents the primary actuator of the MRO xApp. This inclusion ensures completeness of the control-to-KPI mapping and preserves explainability for future scenarios and conflict detection.

Impact of FIR To validate the effectiveness of the proposed scheme in FIR, we conduct an experiment by disabling SA-RAG and β-Reasoning Cascade. As illustrated in Fig. 12, full ORCA reduces the cosine error and NRMSE by 47.0% and 32.5%, respectively, compared to when SA-RAG is disabled. Even without SA-RAG, the LLM agent can still produce feature-importance estimates; however, cosine error and NRMSE increase, indicating less reliable estimates. In contrast, when the β-Reasoning Cascade is removed, the performance closely resembles that of the pure LLM agent. This is primarily because, without β-reasoning, the actively generated queries are overly abstract, making it difficult to retrieve sufficiently relevant knowledge.

*4) **Reasoning of FIR**:* We present a partial response from the FIR module at step 3 in Fig. 13. This example not only demonstrates the internal reasoning process of the LLM agent but also shows how SA-RAG is utilized to ground the reasoning with external knowledge. The first part explains the LLM's logical derivation, followed by the application of SA-RAG for retrieving relevant supporting information.

V. Related Works

A. Conflict mitigation in ORAN

Some DRL-based approaches adopt multi-agent deep Q-learning to foster cooperation among xApps. While effective in controlled settings, they face scalability issues and assume xApps can be retrained jointly, which is impractical in modular, vendor-agnostic O-RAN. xApp Distillation [9] mitigates this by merging pre-trained xApps via policy distillation into a single DQN-based xApp, but this sacrifices modularity and requires re-distillation as conditions evolve. The scheduler in [10] selects xApps using A2C based on current environments but assumes all possible environments are known during training, thus limiting generalization.

The framework in [11] classifies conflicts as direct, indirect, or implicit and resolves them via a Conflict Resolution agent that prioritizes actions. Extensions such as [12], [13] optimize parameter configurations based on utility mappings from KPIs but depend on fixed-environment data, reducing robustness to change. Statistical methods collect ECDFs under conditioned environments [1], using distance metrics to assess conflict severity and deployment compatibility. Others cluster cells by configuration and traffic patterns to predict KPI impacts and preemptively cancel harmful requests [2]. However, these approaches rely heavily on prior environmental assumptions, limiting their applicability in dynamic or unseen scenarios.

B. LLMs for communication network

LLMs have shown strong effectiveness across diverse tasks, including reasoning, embodied decision-making, and code generation. They are now applied across a wide range of disciplines including engineering, computer systems and social science [14]–[21]. Recent research has begun exploring how LLMs can empower communication networks. Early studies focus on tasks like technical question answering over telecom standards, classification of working groups, and document tagging [22]–[29]. In parallel, LLM agents have emerged, enabling models to reason through complex problems using intermediate steps, external tools, or symbolic representations. They show strong potential in dynamic environments, especially for autonomous software development [30]–[36]. Recent works [37], [38] employed LLM agents for network protocol design. Unlike this line of research, we elicit the agent's reasoning capability via a guided multi-step inference enabled by a β-Reasoning Cascade and SA-RAG, allowing it to actively analyze and reason about system behavior with well-grounded domain knowledge.

VI. Conclusion

We present ORCA, the first framework leveraging generative LLM agents to detect and mitigate xApp conflicts in ORAN, achieving up to 60.7% lower cosine error and 65.1% lower NRMSE than baselines. By combining symbolic reasoning with domain knowledge, ORCA delivers fine-grained conflict mitigation and showcases the potential of LLM-driven reasoning for future network optimization.

Acknowledgement

This work was supported in part by a Samsung Fellowship, Samsung Research America, and the UCSD Center for Wireless Communications. The authors would like to express their gratitude to Dr. Hao Chen and other colleagues at Samsung Research America for their insightful feedback and guidance, which significantly enhanced the quality of this paper.

REFERENCES

[1] P. B. del Prever, S. D'Oro, L. Bonati, M. Polese, M. Tsampazi, H. Lehmann, and T. Melodia, "Pacifista: Conflict evaluation and management in open ran," *IEEE Transactions on Mobile Computing*, 2025.

[2] J. Armstrong, E. Fallon, and S. Fallon, "Pre-emptive conflict detection architecture for o-ran service management and orchestration," in *2024 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pp. 335–340, IEEE, 2024.

[3] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[4] "O-ran conflict mitigation," Tech. Rep. O-RAN.WG3.TR.ConMit-R004-v01.00, O-RAN Alliance, Working Group 3, February 2024. Technical Report.

[5] L. S. Shapley *et al.*, "A value for n-person games," 1953.

[6] O-RAN Alliance, "O-RAN Architecture Description, v10.00," 2023. https://www.o-ran.org/specifications.

[7] OpenAI, "gpt-4o-mini." https://platform.openai.com, 2025. Accessed via OpenAI API.

[8] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[9] H. Erdol, X. Wang, R. Piechocki, G. Oikonomou, and A. Parekh, "xapp distillation: Ai-based conflict mitigation in b5g o-ran," *arXiv preprint arXiv:2407.03068*, 2024.

[10] I. Cinemre and T. Mahmoodi, "xapp conflict mitigation with scheduler," *arXiv preprint arXiv:2504.06867*, 2025.

[11] C. Adamczyk and A. Kliks, "Conflict mitigation framework and conflict detection in o-ran near-rt ric," *IEEE Communications Magazine*, vol. 61, no. 12, pp. 199–205, 2023.

[12] A. Wadud, F. Golpayegani, and N. Afraz, "Conflict management in the near-rt-ric of open ran: A game theoretic approach," in *2023 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pp. 479–486, IEEE, 2023.

[13] A. Wadud, F. Golpayegani, and N. Afraz, "Qacm: Qos-aware xapp conflict mitigation in open ran," *IEEE Transactions on Green Communications and Networking*, 2024.

[14] Y. Qin and Liang, "Toolllm: Facilitating large language models to master 16000+ real-world apis," *arXiv preprint arXiv:2307.16789*, 2023.

[15] M. Safdari, G. Serapio-García, C. Crepy, S. Fitz, P. Romero, L. Sun, M. Abdulhai, A. Faust, and M. Matarić, "Personality traits in large language models," *arXiv preprint arXiv:2307.00184*, 2023.

[16] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[17] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang, *et al.*, "Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory," *arXiv preprint arXiv:2305.17144*, 2023.

[18] B. Hu, C. Zhao, P. Zhang, Z. Zhou, Y. Yang, Z. Xu, and B. Liu, "Enabling intelligent interactions between an agent and an llm: A reinforcement learning approach," *arXiv preprint arXiv:2306.03604*, 2023.

[19] C. Ziems, W. Held, O. Shaikh, J. Chen, Z. Zhang, and D. Yang, "Can large language models transform computational social science?," *Computational Linguistics*, vol. 50, no. 1, pp. 237–291, 2024.

[20] J. J. Horton, "Large language models as simulated economic agents: What can we learn from homo silicus?," tech. rep., National Bureau of Economic Research, 2023.

[21] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023.

[22] H. Holm, "Bidirectional encoder representations from transformers (bert) for question answering in the telecom domain.: Adapting a bert-like language model to the telecom domain using the electra pre-training approach," 2021.

[23] A. Karapantelakis, M. Shakur, A. Nikou, F. Moradi, C. Orlog, F. Gaim, H. Holm, D. D. Nimara, and V. Huang, "Using large language models to understand telecom standards," *arXiv preprint arXiv:2404.02929*, 2024.

[24] L. Bariah, H. Zou, Q. Zhao, B. Mouhouche, F. Bader, and M. Debbah, "Understanding telecom language through large language models," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*, pp. 6542–6547, IEEE, 2023.

[25] N. Piovesan, A. De Domenico, and F. Ayed, "Telecom language models: Must they be large?," *arXiv preprint arXiv:2403.04666*, 2024.

[26] A. Maatouk, F. Ayed, N. Piovesan, A. De Domenico, M. Debbah, and Z.-Q. Luo, "Teleqna: A benchmark dataset to assess large language models telecommunications knowledge," *arXiv preprint arXiv:2310.15051*, 2023.

[27] H. Cai and S. Wu, "Tkg: Telecom knowledge governance framework for llm application," 2023.

[28] G. Charan, M. Alrabeiah, and A. Alkhateeb, "Vision-aided 6g wireless communications: Blockage prediction and proactive handoff," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10193–10208, 2021.

[29] M. Matsuura, Y. K. Jung, and S. N. Lim, "Visual-llm zero-shot classification," 2023.

[30] Y. Dong, X. Jiang, Z. Jin, and G. Li, "Self-collaboration code generation via chatgpt," *arXiv preprint arXiv:2304.07590*, 2023.

[31] W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C. Qian, C.-M. Chan, Y. Qin, Y. Lu, R. Xie, *et al.*, "Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents," *arXiv preprint arXiv:2308.10848*, 2023.

[32] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, *et al.*, "Metagpt: Meta programming for multi-agent collaborative framework," *arXiv preprint arXiv:2308.00352*, 2023.

[33] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, and H. Ji, "Unleashing cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration," *arXiv preprint arXiv:2307.05300*, vol. 1, no. 2, p. 3, 2023.

[34] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun, "Communicative agents for software development," *arXiv preprint arXiv:2307.07924*, 2023.

[35] D. Huang, Q. Bu, J. M. Zhang, M. Luck, and H. Cui, "Agentcoder: Multi-agent-based code generation with iterative testing and optimisation," *arXiv preprint arXiv:2312.13010*, 2023.

[36] G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: Communicative agents for" mind" exploration of large language model society," *Advances in Neural Information Processing Systems*, vol. 36, pp. 51991–52008, 2023.

[37] D. C. Kwon and X. Zhang, "CP-AgentNet: Autonomous and Explainable Communication Protocol Design Using Generative Agents," in *Proceedings of IEEE ICNP*, 2025.

[38] Z. He, A. Gottipati, L. Qiu, X. Luo, K. Xu, Y. Yang, and F. Y. Yan, "Designing network algorithms via large language models," in *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, pp. 205–212, 2024.