Intent-to-Learning Translation for Computing Continuum Management

Cveta Capova
Distributed Systems Group
TU Wien, Austria
c.capova@dsg.tuwien.ac.at

Andrea Morichetta
Distributed Systems Group
TU Wien, Austria
a.morichetta@dsg.tuwien.ac.at

Anna Lackinger
Distributed Systems Group
TU Wien, Austria
a.lackinger@dsg.tuwien.ac.at

Schahram Dustdar
Distributed Systems Group
TU Wien, Austria
dustdar@dsg.tuwien.ac.at

2025 IEEE 33rd International Conference on Network Protocols (ICNP) | 979-8-3315-0376-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICNP65844.2025.11192441

***Abstract*—This paper offers a solution to generate concrete goals for automating the fulfillment of user intents in the computing continuum. The computing continuum guarantees a flexible infrastructure for services at the cost of more complex handling. Our proposed method innovates the state of the art, helping build performative automated strategies through the translation of service owner intents into concrete targets. We improve on existing intent-based systems by offering support for multi-domain infrastructures. Furthermore, we go beyond current computing continuum management solutions, offering full automation by generating concrete targets for the continuum of automated agents. We achieve that through a multi-agent system built on Large Language Models (LLMs) that translates high-level business intents into executable Reinforcement Learning (RL) environments. By leveraging infrastructure representations in the form of Knowledge Graphs, the framework identifies which system components require adaptation and estimates the target metric values needed to fulfill the intent. We evaluate it on a realistic use case with promising results. We can deploy a fully working RL agent to manage network and computing resources, achieving a success rate higher than 80% after preliminary training.**

***Index Terms*—intent translation, reward generation, reinforcement learning, computing continuum, resource management**

I. Introduction

The computing continuum allows the deployment of applications on the whole infrastructure spectrum, from the smallest nodes to the cloud and from computing to network components, allowing extreme flexibility. However, this design significantly increases the effort required to maintain and manage it [1]. Application owners must ensure the correct functionality of their services and address non-functional requirements such as availability, latency, and cost in a wider and much more diverse platform. This scenario requires vast expert knowledge, which is burdensome and expensive to obtain. For example, smart city scenarios have strict real-time demands, requiring complex optimization of the whole computing continuum. In contrast, some services may instead prioritize cost over latency. Therefore, a mechanism is needed to evaluate and translate the application owner's goals and adapt the system architecture according to them.

Intent-based management aims at addressing this challenge [2]. In this approach, users define high-level objectives, and the system autonomously interprets them and adapts its configuration to meet these goals. The main advantage of this paradigm is that it allows users to specify desired outcomes (such as secure or reliable infrastructures) without having to understand low-level details [3]. Current efforts extend the intent paradigm, initially developed in the network domain, to computing. [1], [4]–[6] All these solutions also address the mission to develop strategies for enforcing the application owners' intents on the infrastructure. Typically, they rely on Reinforcement Learning (RL) algorithms to centrally instruct autoscaling or scheduling actions. In contrast, recent research [7]–[10] envisions a distributed approach, where the computing continuum is partitioned into logical or geographical groups, each of which autonomously makes decisions on its infrastructure slice. Such an approach calls for a middleware, i.e., a coordinator, that ensures that each of these partitions can work successfully to fulfill the intents. This way, it is possible to achieve a modular and flexible way of managing the continuum; each infrastructure instance, whether it is computing, network, or storage, can use its agents or models. The coordinator connects them with concrete objectives that guarantee the application's intent overall. Still, both the coordinator and the infrastructure agents need to have concrete goals for developing their strategies. Whether the target is an RL algorithm or something else, this task requires manual configuration, making it complex to have an autonomous and scalable system management.

Our approach addresses the current gap by proposing a method to convert business-level intents into actionable learning objectives suitable for such coordination tools. We achieve this by composing a multi-agent, LLM-based system for information gathering and reasoning. We enrich the LLM with domain-specific data in the form of Knowledge Graphs (KGs) that represent the continuum infrastructure and the target application, through Retrieval-Augmented Generation with Graphs (GraphRAG) processes. In particular, given RL's broad adoption, we focus on providing an RL environment as the output of the intent translation process. This method returns service-relevant metrics as the observable state space, possible system adjustments as the action space, and the user's custom intent as the reward function. In doing so, the full scope of the manageable infrastructure—including compute, network, and storage resources—is taken into account, along with its continuous improvement through learning in dynamic

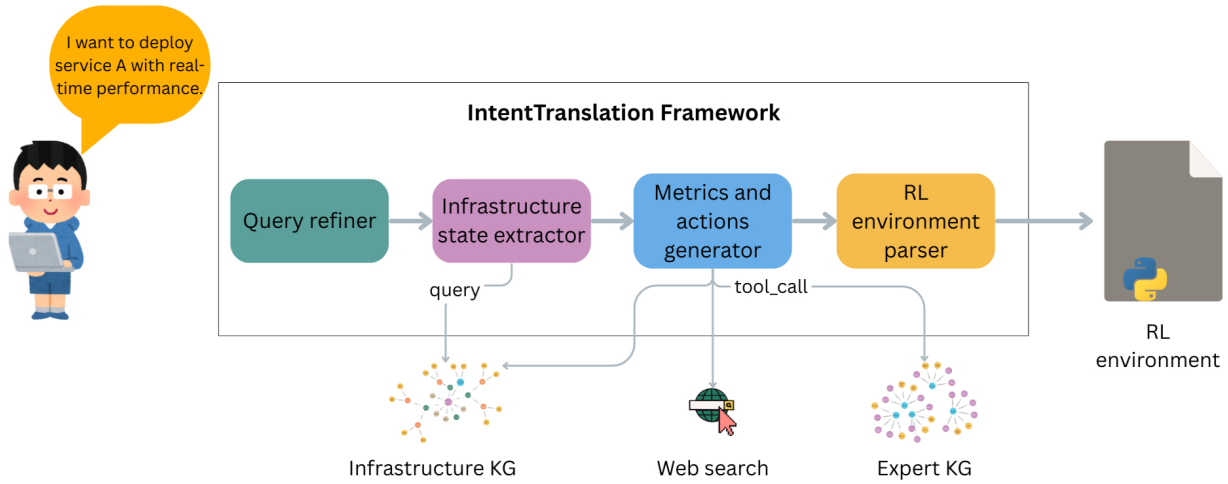


Fig. 1: Workflow overview for translating high-level intent into an RL environment

environments. By testing it in a UAV-based use case, we show encouraging results. We are able to generate a working, turnkey environment for the RL agent. Furthermore, the approach is able to satisfy the intent for more than 80% of the cases. Significantly, we show how we can learn, even with little information from the infrastructure instances, goals, and strategies for managing a multi-domain continuum.

To summarize, our main contribution involves addressing the gap of how to instrument the computing continuum management algorithms by **(I)** developing a multi-agent, LLM-based method for translating high-level intents into actionable RL environments. Furthermore, **(II)** we extend the state-of-the-art by offering a coordinator for multi-domain infrastructure, where the infrastructure agents act autonomously in their environment. Finally, **(III)**, we show how this paradigm can work empirically, without requiring low-level details on the infrastructure components, displaying promising outcomes.

II. Methodology

In this section, we introduce the workflow for translating a user's high-level intent into an RL environment, illustrated in Fig. 1.

A. Knowledge Graphs

Knowledge Graphs (KGs) are structured representations of information expressed as subject–predicate–object triples. The Knowledge Graphs utilize nodes and edges to model complex relationships and uncover hidden dependencies, thereby enabling both efficient querying and flexible data modeling. In our proposed framework, we incorporate Retrieval-Augmented Generation with Graphs (GraphRAG) to integrate the structured knowledge from KGs into the reasoning process of Large Language Models (LLMs). GraphRAG extends the standard Retrieval-Augmented Generation (RAG) approach—commonly used to supplement LLMs with external context retrieved from documents, databases, images, and other sources—by leveraging graph-based knowledge representations. This integration allows the LLM to generate responses based not only on its pre-trained parameters but also on dynamic, contextually relevant information from the KG, leading to improved accuracy and reliability in generated outputs [11]. For graph storage and querying, we adopt Neo4j, due to its compatibility with LangGraph[1] and LangChain[2] frameworks, and its support for expressive and LLM-friendly reasoning queries. This choice is further supported by findings in [12], which indicate that Neo4j-based reasoning and query generation are more interpretable for LLMs compared to RDF-based graph structures.

As part of our approach, we employ two distinct types of knowledge graphs (KGs): the Infrastructure Knowledge Graph and the Expert Knowledge Graph. The Infrastructure KG captures the current state of the user's cloud infrastructure. It encodes information about available computational resources and deployed applications, represented as nodes, with edges denoting their logical or physical connections (e.g., `(:Service)-[:RUNS_ON]->(:K8sCluster)`). Each application or infrastructure node is linked to `:Metric` nodes, which represent key observable metrics and their current values, such as CPU usage, memory utilization, and service or network latency. In addition, each metric node has an associated boolean parameter indicating whether its value can be adapted and set as a target value for intent-based management tools. The *Expert KG* encapsulates domain knowledge to guide the reasoning process of the LLM agents. It includes information such as dependencies between Service Level Indicators (SLIs) or history of recommended actions on previous intents.

B. Translation workflow

An overview of the translation workflow is shown in Fig.2. The process is based on a multi-agent framework, where each stage is handled by a specialized LLM agent to provide more reliable and task-specific results. For cost

[1]https://www.langchain.com/langgraph
[2]https://www.langchain.com/

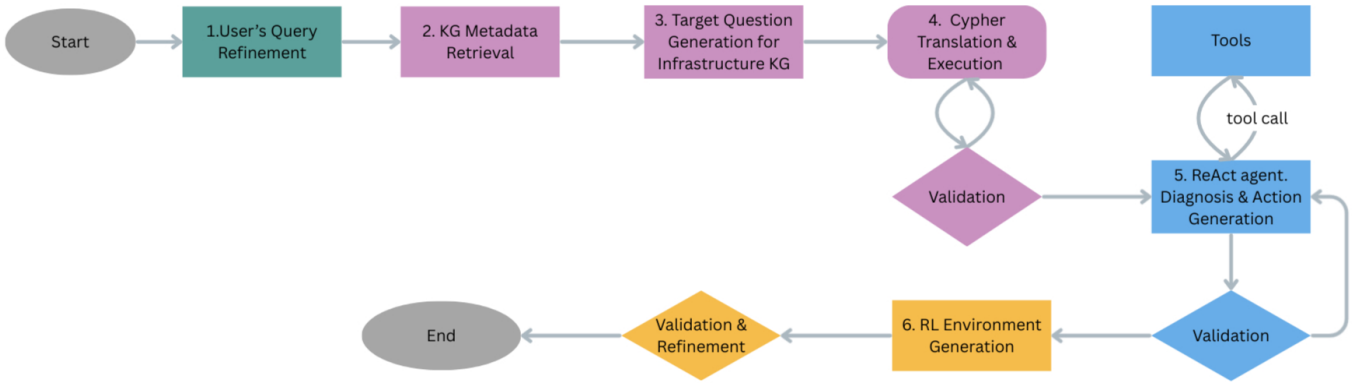


Fig. 2: Detailed translation workflow

efficiency, `gpt-4o-mini` was used in all stages except Step 5 (Diagnosis & Action Generation) and Step 6 (RL Environment Generation). In these two stages, `gpt-4.1` was chosen for its better reasoning, reliable tool invocation, and broad cross-domain knowledge, which are essential for producing consistent diagnostic actions and valid RL environment specifications. To coordinate the interactions among these agents, we utilize LangGraph due to its stateful orchestration capabilities and fine-grained control over agent execution flow and communication.

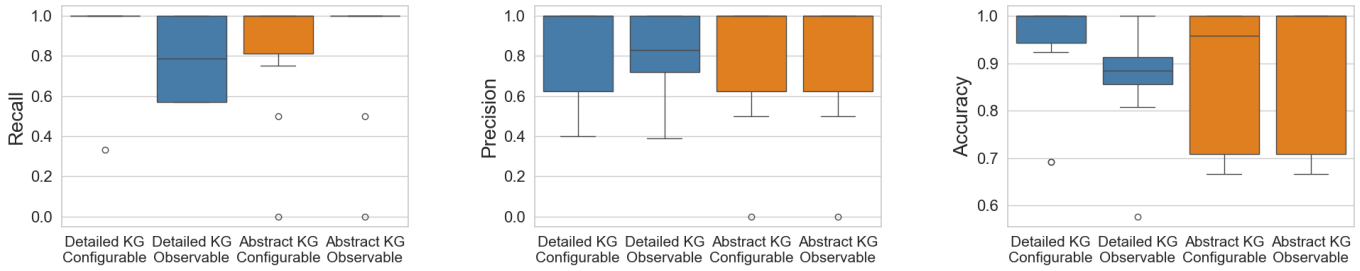
The translation process begins with a high-level business intent provided by the user in natural language. Following the approach used in [13], we break down the complex tasks into manageable subtasks for improved reasoning and modularity. The initial input is handled by a query-refiner agent, which decomposes the vague, high-level request into a set of more specific sub-questions that guide further research.

Using the refined sub-questions and the schema of the Infrastructure KG, the system identifies which node labels and relationship types are relevant for answering the user's intent (Step 2) and generates natural language queries for the KG (Step 3). Each of these queries is then translated into a Cypher query using the GraphCypherQAChain module from the LangChain library, which supports natural-language-to-Cypher translation over a predefined graph schema. Then a validation agent verifies the completeness of the extracted relevant system state and, if necessary, generates additional questions to retrieve any missing data. This multi-step refinement is essential. Attempting to directly convert a high-level intent into a single Cypher query often results in either overly broad queries that retrieve irrelevant information or overly narrow ones that omit crucial context. The staged decomposition enables more precise extraction of relevant system state data from the Infrastructure KG.

Once the relevant system state is extracted, a domain-specific ReAct agent—designed to impersonate a cloud-continuum-management expert—is invoked. The agent analyzes the retrieved context and proposes corrective or optimization actions, taking into account system constraints and configurable parameters. Implemented according to the ReAct paradigm, the agent combines reasoning with action: the LLM generates intermediate reasoning steps and calls external tools to support its decisions [14]. In our implementation, the agent can invoke a web-search tool to retrieve additional, up-to-date information and a GraphRAG interface over the Infrastructure and Expert KGs to obtain verified domain knowledge. This iterative loop continues until the agent gathers sufficient information to formulate a response or reaches a predefined timeout. The final output is a textual explanation of the problem's cause, recommended adaptation actions, and estimated target values for key metrics, derived from the Infrastructure KG. Once this response is generated, a validation agent reviews it for correctness and soundness. If the response is found to be insufficient, the process loops back for further refinement. Otherwise, the validated recommendation is passed to an agent responsible for translating it into executable code for a reinforcement learning environment—specifically, a Gymnasium-compatible format, which ensures better interoperability and standardization [15]. Utilizing the previously extracted system state, the agent constructs an observation space composed of metrics relevant to intent fulfillment. It also defines the action space based on the identified configurable parameters, enabling their adaptation while enforcing user-defined constraints. Moreover, the agent formulates a reward function that aligns with the user's business objectives. The final step of the translation workflow is a second validation phase, where an agent evaluates the generated code and refines it to ensure the correctness and reliability of the program.

III. Evaluation

We evaluate our translation workflow in a distributed drone image classification scenario. In this setting, drones transmit images to a Kubernetes cluster that hosts several services, including object detection, black-and-white conversion, image resizing, an alarm system, and a database service. The cluster consists of one master node and three worker nodes, with each service assigned to a dedicated host. Furthermore, each host is connected to a network switch, which is part of the overall network topology. Based on this infrastructure, we created two knowledge graphs. The first KG includes detailed



(a) Recall (b) Precision (c) Accuracy

Fig. 3: Recall, precision and accuracy for the retrieved relevant metrics

metrics and infrastructure information such as network and service latency, service CPU limits, host CPU usage, service memory limits, and concrete network topology. The second KG is more abstract and includes only high-level metrics, such as service and network latency and throughput. In both KGs, metrics are modeled as separate nodes, and for the scope of this work, their values are static, i.e., not derived from live measurements. Furthermore, we distinguish between observable and configurable metrics. A comparison of the two KGs in terms of these metric types is shown in Table I. Configurable metrics have the parameter `tunable` set to `true`, indicating that they can be adjusted to meet estimated values. Observable metrics, by contrast, are not part of the action space; instead, they are used in the observation state, the reward function, or to support the reasoning process during the translation workflow. A key aspect of evaluating the translation workflow is assessing whether the system successfully retrieves the relevant metrics for the given task and correctly distinguishes between observable and configurable metrics. It is worth noting that the number of configurable metrics is the same in both the detailed and abstract KGs. Furthermore, in the abstract KG, all observable metrics are also configurable. In our scenario, configurable metrics include network latency, network throughput, and latency and throughput for each service. For the translation workflow to be effective, it must recognize the service that requires adaptation and take into consideration both networking and computing resource constraints.

TABLE I: Comparison of metrics in Abstract vs. Detailed KG

Type	**Abstract KG**	**Detailed KG**
Observable Metrics	Service Latency, Service Throughput, Network Latency, Network Throughput	Service Latency, Service Throughput, Service CPU Limit, Service Memory Limit, K8S Host CPU Usage, Network Latency, Network Throughput
Configurable Metrics	Service Latency, Service Throughput, Network Latency, Network Throughput	Service Latency, Service Throughput, Network Latency, Network Throughput

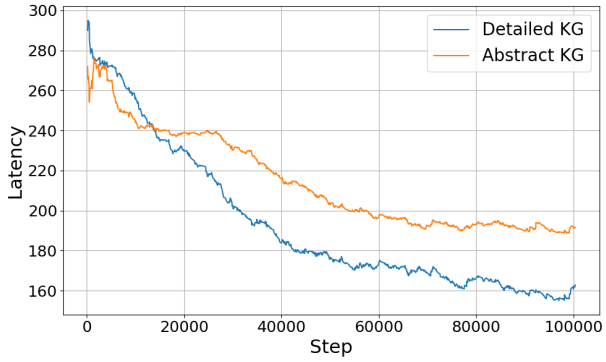
To evaluate its performance, we ran 10 times the translation workflow for the detailed KG and the abstract KG. In this scenario, the user expresses a real-time performance requirement for the object detection service in natural language:

"*I am going to fly my drone in Vienna, and I need to be sure that object detection classifications of people are sent to my dashboard in real time without loss.*"

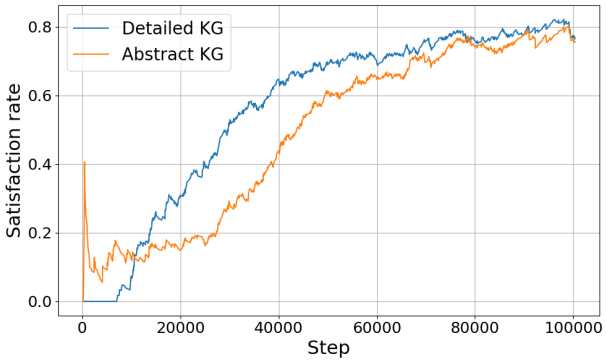
As illustrated in Fig. 3a, our translation system demonstrates high recall, indicating that most of the relevant configurable or observable metrics are correctly identified and included in the RL action space. Notably, the recall for retrieving relevant configurable metrics is higher when using the detailed KG compared to the abstract KG, suggesting that the presence of more detailed metrics facilitates better identification of relevant ones. In contrast, the recall for retrieving observable metrics from the detailed KG is comparatively lower. This is because the detailed KG contains a larger number of additional observable metrics than the abstract KG. These additional metrics are not strictly required to fulfill the system's operational intent and instead serve to complement configurable metrics by enriching the observation space, so the agent does not recognize them as equally important.

Fig. 3b shows the system's precision, reflecting its ability to filter out irrelevant metrics. The observed variance in precision is primarily due to cases where the agent mistakenly retrieved latency and throughput values for all services instead of restricting them to the object detection service relevant to the intent. Despite this, the system achieves consistently high accuracy across all configurations (Fig. 3c), where accuracy is defined as the proportion of relevant metrics correctly identified as true positives (TP) and irrelevant metrics correctly excluded as true negatives (TN) out of the total metrics evaluated. With most values clustered around 0.8, these results indicate generally reliable identification of metrics for selection into the generated RL environment's action and observation spaces.

To evaluate the ability of the generated RL environments to learn and converge toward optimal behavior, we selected one environment for each knowledge graph and trained them for 100,000 steps. As illustrated in Fig. 4a, both environments effectively capture the user's intent for real-time performance. Through dynamic adaptation of network and service latencies, the agents progressively reduce the overall end-to-end latency over time. To assess how well the agents meet the required



(a) Total latency over time



(b) Constraints satisfaction rate

Fig. 4: RL environment evaluation

constraints during training, we introduce the *constraint satisfaction rate* metric, which reflects the agent's ability to consistently fulfill the system's requirements defined within the RL reward function. At each timestep, a binary value is recorded: 1 if all constraints are met, and 0 otherwise. The satisfaction rate is then computed as the average of these values over a sliding window of recent rollouts (e.g., the last 30). An example reward function used to enforce these latency and throughput constraints is shown in Fig. 5. It penalizes violations of individual constraints—such as exceeding service or network latency thresholds, or dropping below minimum throughput levels—while also rewarding full constraint compliance.

As shown in Fig. 4b, the RL agents demonstrate the ability to reliably meet all defined constraints in the environment. This result is particularly significant, as it indicates that latency reductions are achieved within the specified resource boundaries—avoiding both overprovisioning and underprovisioning of system components.

IV. Related Work

A. Reward Function Generation in RL

Since the reward function is one of the core components of the RL environment generated by our solution, designing an effective reward is crucial for enabling agents to learn behaviors that fulfill the user's intent. In this section, we review

```python
def calculate_reward(self, s_latency, n_latency,
    s_through, n_through, end_to_end_latency):
        reward = 0.0

        if end_to_end_latency >
           self.hard_constraint_latency:
            reward -= 50.0

        reward += -0.1 * max(s_latency - 150.0, 0)
        reward += -0.1 * max(n_latency - 100.0, 0)
        reward += -2.0 * max(30.0 - s_through, 0)
        reward += -2.0 * max(1.0 - n_through, 0)
        reward += 2.0 * (s_through >= 30.0)
        reward += 2.0 * (n_through >= 1.0)

        if (s_latency <= 150.0 and
            n_latency <= 100.0 and
            s_through >= 30.0 and
            n_through >= 1.0 and
            end_to_end_latency <=
            self.hard_constraint_latency):
            reward += 10.0

        return reward
```

Fig. 5: Reward function generated by the translation workflow using the Abstract KG

prior work aimed at automating the design of reward functions. One foundational approach is Inverse Reinforcement Learning (IRL) [16], which infers reward functions from expert demonstrations rather than manually specifying them. While IRL has been influential, its effectiveness is often limited by the quality and availability of expert data [17]. Recent work has explored leveraging large language models (LLMs) to address these limitations. Xie et al. [18] introduce a framework that generates reward functions from natural language task descriptions and structured Python environment specifications, incorporating human-in-the-loop feedback to refine reward quality. Sun et al. [19] extend this idea by introducing an automated evaluation module, which eliminates the need for human refinement by assessing reward quality through agent performance and task success metrics. Further, Wang et al. [20] propose a system that jointly generates reward functions and state representations from natural language. These are used during the RL training, and episode returns serve as feedback for the LLM.

B. LLM Reasoning and Structured Knowledge Integration

Reasoning with LLMs is central to our translation workflow, particularly in interpreting structured knowledge. A large body of research focuses on enhancing LLM reasoning through prompting techniques, including in-context learning, chain-of-thought (CoT), and few-shot prompting [21]. Role-play prompting [22] further enriches this paradigm by enabling LLMs to adopt diverse perspectives, thereby improving context-aware reasoning. Extending this line of research, multi-agent architectures have been explored, where LLMs with specialized roles collaborate to solve complex tasks [23]. These frameworks are often enhanced with tool-augmented

agents [24], capable of web search, computation, or domain-specific operations. Wu et al. [25] demonstrate such a system in the context of multi-step reasoning, combining tool-use with KG-based mind maps to guide deductive processes. Li et al. [26] similarly integrate LLMs with knowledge graphs for question answering, allowing the model to extract task-relevant subgraphs that augment the prompt with structured contextual information.

While prior work explores reward synthesis from language and structured data, it largely focuses on textual environments or robotic control scenarios. In contrast, our approach introduces a framework that constructs full RL environments from domain-specific knowledge graphs, targeting cloud–edge continuum management—a setting underexplored in existing literature. Our method builds upon the multi-agent and GraphRAG strategies of [25] and [26], respectively, but extends them beyond answer generation. We integrate KG-guided reasoning with LLM-driven translation to extract not only reward functions but also observation and action spaces. This results in executable RL environments tailored to infrastructure management intents, bridging structured domain knowledge and autonomous learning workflows.

V. Conclusions

In this article, we introduced a workflow for translating high-level business objectives into RL environment files tailored for computing continuum management. We demonstrated how, based on a provided infrastructure Knowledge Graph, a user's intent—expressed in natural language—can be mapped to concrete resource-level metrics and used to drive resource-level adaptation actions, while preserving the global objective. Future work will focus on enhancing the dynamic updating of the infrastructure KG to better capture real-time metrics and system state changes. We also plan to enable automatic extraction of KG triples from LLM outputs, supporting continuous enrichment of the expert knowledge graph. Lastly, we aim to validate our approach beyond simulated settings by applying our RL environment files in real-world infrastructure scenarios, showcasing the capabilities of our approach in practice.

VI. Acknowledgment

This work is funded by the HORIZON Research and Innovation Action 101135576 INTEND "Intent-based data operation in the computing continuum."

References

[1] N. Akbari, J. Grundy, A. Cheema, and A. N. Toosi, "Intentcontinuum: Using llms to support intent-based computing across the compute continuum," *arXiv preprint arXiv:2504.04429*, 2025.

[2] A. Leivadeas and M. Falkner, "A survey on intent-based networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 625–655, 2022.

[3] L. Pang, C. Yang, D. Chen, Y. Song, and M. Guizani, "A survey on intent-driven networks," *IEEE Access*, vol. 8, pp. 22862–22873, 2020.

[4] T. Metsch, M. Viktorsson, A. Hoban, M. Vitali, R. Iyer, and E. Elmroth, "Intent-driven orchestration: Enforcing service level objectives for cloud native deployments," *SN Computer Science*, vol. 4, no. 3, p. 268, 2023.

[5] A. Morichetta, N. Spring, P. Raith, and S. Dustdar, "Intent-based management for the distributed computing continuum," in *2023 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 239–249, IEEE, 2023.

[6] N. Filinis, I. Tzanettis, D. Spatharakis, E. Fotopoulou, I. Dimolitsas, A. Zafeiropoulos, C. Vassilakis, and S. Papavassiliou, "Intent-driven orchestration of serverless applications in the computing continuum," *Future Generation Computer Systems*, vol. 154, pp. 72–86, 2024.

[7] A. Morichetta, V. C. Pujol, and S. Dustdar, "A roadmap on learning and reasoning for distributed computing continuum ecosystems," in *2021 IEEE International Conference on Edge Computing (EDGE)*, pp. 25–31, IEEE, 2021.

[8] A. Morichetta, A. Lackinger, and S. Dustdar, "Cohabitation of intelligence and systems: Towards self-reference in digital anatomies," in *2024 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 102–110, IEEE, 2024.

[9] D. Firmani, F. Leotta, J. G. Mathew, J. Rossi, L. Balzotti, H. Song, D. Roman, R. Dautov, S. Sen, V. Balionyte-Merle, *et al.*, "Intend: Intent-based data operation in the computing continuum," in *CEUR Workshop Proceedings*, vol. 3692, pp. 43–50, CEUR-WS, 2024.

[10] A. Morichetta, J. Brenes, M. Kolobov, D. Dib, T. Metsch, A. Lackinger, C. Capova, H. Song, R. Dautov, A. Khalid, and S. Dustdar, "incoord: Intent-based coordination in the multi-domain cloud-edge continuum," in *2025 IEEE 33rd International Conference on Network Protocols (ICNP)*, IEEE, 2025.

[11] Q. Zhang, S. Chen, Y. Bei, Z. Yuan, H. Zhou, Z. Hong, J. Dong, H. Chen, Y. Chang, and X. Huang, "A survey of graph retrieval-augmented generation for customized large language models," *arXiv preprint arXiv:2501.13958*, 2025.

[12] Y. Feng, S. Papicchio, and S. Rahman, "Cypherbench: Towards precise retrieval over full-scale modern knowledge graphs in the llm era," *arXiv preprint arXiv:2412.18702*, 2024.

[13] Z. Chen, K. Liu, Q. Wang, J. Liu, W. Zhang, K. Chen, and F. Zhao, "Mindsearch: Mimicking human minds elicits deep ai searcher," *arXiv preprint arXiv:2407.20183*, 2024.

[14] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[15] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulao, A. Kallinteris, M. Krimmel, A. KG, *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.

[16] A. Y. Ng, S. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *Icml*, vol. 1, p. 2, 2000.

[17] Y. Zeng, Y. Mu, and L. Shao, "Learning reward for robot skills using large language models via self-alignment," *arXiv preprint arXiv:2405.07162*, 2024.

[18] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu, "Text2reward: Reward shaping with language models for reinforcement learning," *arXiv preprint arXiv:2309.11489*, 2023.

[19] S. Sun, R. Liu, J. Lyu, J.-W. Yang, L. Zhang, and X. Li, "A large language model-driven reward design framework via dynamic feedback for reinforcement learning," *arXiv preprint arXiv:2410.14660*, 2024.

[20] B. Wang, Y. Qu, Y. Jiang, J. Shao, C. Liu, W. Yang, and X. Ji, "Llm-empowered state representation for reinforcement learning," *arXiv preprint arXiv:2407.13237*, 2024.

[21] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, *et al.*, "The prompt report: A systematic survey of prompting techniques," *arXiv preprint arXiv:2406.06608*, vol. 5, 2024.

[22] A. Kong, S. Zhao, H. Chen, Q. Li, Y. Qin, R. Sun, X. Zhou, E. Wang, and X. Dong, "Better zero-shot reasoning with role-play prompting," *arXiv preprint arXiv:2308.07702*, 2023.

[23] Y. Talebirad and A. Nadiri, "Multi-agent collaboration: Harnessing the power of intelligent llm agents," *arXiv preprint arXiv:2306.03314*, 2023.

[24] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, *et al.*, "The rise and potential of large language model based agents: A survey," *Science China Information Sciences*, vol. 68, no. 2, p. 121101, 2025.

[25] J. Wu, J. Zhu, and Y. Liu, "Agentic reasoning: Reasoning llms with tools for the deep research," *arXiv preprint arXiv:2502.04644*, 2025.

[26] Y. Li, R. Zhang, and J. Liu, "An enhanced prompt-based llm reasoning scheme via knowledge graph-integrated collaboration," in *International Conference on Artificial Neural Networks*, pp. 251–265, Springer, 2024.