RAILS: Risk-Aware Iterated Local Search for Joint SLA Decomposition and Service Provider Management in Multi-Domain Networks

Cyril Shih-Huan Hsu
Informatics Institute
University of Amsterdam
1098 XH Amsterdam, The Netherlands
s.h.hsu@uva.nl

Chrysa Papagianni
Informatics Institute
University of Amsterdam
1098 XH Amsterdam, The Netherlands
c.papagianni@uva.nl

Paola Grosso
Informatics Institute
University of Amsterdam
1098 XH Amsterdam, The Netherlands
p.grosso@uva.nl

***Abstract*—The emergence of the fifth generation (5G) technology has transformed mobile networks into multi-service environments, necessitating efficient network slicing to meet diverse Service Level Agreements (SLAs). SLA decomposition across multiple network domains, each potentially managed by different service providers, poses a significant challenge due to limited visibility into real-time underlying domain conditions. This paper introduces Risk-Aware Iterated Local Search (RAILS), a novel risk model-driven meta-heuristic framework designed to jointly address SLA decomposition and service provider selection in multi-domain networks. By integrating online risk modeling with iterated local search principles, RAILS effectively navigates the complex optimization landscape, utilizing historical feedback from domain controllers. We formulate the joint problem as a Mixed-Integer Nonlinear Programming (MINLP) problem and prove its NP-hardness. Extensive simulations demonstrate that RAILS achieves near-optimal performance, offering an efficient, real-time solution for adaptive SLA management in modern multi-domain networks.**

I. Introduction

The advent of 5G has transformed mobile networks into multi-service environments tailored to diverse industry needs. A key enabler of this shift is network slicing, which creates multiple End-to-End (E2E) logical networks over shared infrastructure, each customized per Service Level Agreements (SLAs). SLAs define expected Quality of Service (QoS) through Service-Level Objectives (SLOs), covering metrics like throughput, latency, reliability, and security. A single network slice may span across multiple segments of the network, including (radio) access, transport, and core networks, and it may involve collaboration between different operators and infrastructure providers. To ensure that the service meets the agreed-upon SLOs across these domains, it is essential to adjust the service parameters accordingly. As a result, the E2E SLA associated with a network slice must be partitioned into specific SLOs for each domain. This decomposition is crucial for effective resource allocation and remains a core challenge in network slicing. Several studies have discussed this issue. [1] highlights the complexity of mapping E2E requirements to transport networks. [2] focuses on lifecycle automation, orchestration, and real-time monitoring for SLA compliance. [3] stresses the role of SLA parameters in E2E QoS and the need for appropriate transport resources. Additionally, [4] underscores the importance of SLA decomposition for resource allocation, while [5] explores AI-assisted SLA decomposition in automating 6G business processes.

In typical network slicing management architectures, a two-level hierarchy is employed [6], [7], [8]. This includes an E2E service orchestrator, responsible for overseeing the lifecycle management of network services, and local domain controllers, which manage the instantiation of network slices within their specific domains. The orchestrator determines how the E2E SLA is partitioned into domain-specific SLOs. However, a common constraint is that the orchestrator usually lacks real-time visibility into the state of each domain's infrastructure at the moment of decomposition. Instead, it relies on historical data reflecting the outcomes of previous slice requests. Several studies [9], [10], [11] have introduced prediction-based approaches for SLA management, though they do not explicitly tackle the E2E SLA decomposition problem. In [9], the authors proposed a mapping layer that oversees the network within a service area, managing radio resource allocation to slices to ensure their target service requirements are met. The work in [10] presented an SLA-constrained optimization method leveraging Deep Learning (DL) to estimate resource requirements based on per-slice traffic. Similarly, [11] utilized a context-aware approach, employing graph representations to predict SLA violations in cloud computing environments. Additionally, heuristic-based SLA decomposition methods have been explored in prior research [4]. In [12], the authors introduced an E2E SLA decomposition system that applies supervised machine learning to partition E2E SLAs into access, transport, and core SLOs.

In our previous work [6], [7], we tackled the SLA decomposition problem using risk models in a two-step approach that combined machine learning and optimization. Building on that, [8] introduced an online learning–decomposition framework for dynamic, multi-domain SLA management. However, these studies assumed a pre-selected service provider per domain, focusing solely on E2E acceptance probabilities. In practical, real-world network environments, multiple service providers are often available within each domain, offering varying performance characteristics and capabilities. For example, in a 5G network slice for autonomous vehicles, Ericsson provides high-capacity RAN for low-latency urban coverage, Nokia ensures reliable transport with energy-efficient networking, and AWS offers a scalable cloud-native core. This combination ensures stringent SLA requirements for real-time communication. As a result, the optimization process should consider both the decomposition of SLAs and the selection of providers across domains to ensure more flexible and efficient resource utilization in multi-domain networks. To address these limitations, this paper introduces Risk-Aware Iterated Local Search (RAILS), a novel risk model-driven meta-heuristic framework. RAILS extends the principles of Iterated Local Search (ILS) by integrating dynamic risk modeling and SLA decomposition techniques proposed in [8]. The main contributions of this paper are:

1. We formulate the joint SLA decomposition and service provider selection tasks as a Mixed-Integer Nonlinear Programming (MINLP) problem and demonstrate its NP-hardness.
2. We propose RAILS, a novel risk-aware meta-heuristic framework designed to jointly address the SLA decomposition and service provider selection problem.
3. We empirically show that RAILS achieves near-optimal performance within an analytic model-based simulation environment with low computational time.

The paper is organized as follows: Section II defines the system model and formulates the problem. Section III introduces the RAILS framework. Section IV details the simulation setup, while Section V presents and discusses the results. Section VI concludes the paper.

II. System Model

A. Problem Formulation

Let N denote the number of domains that the service spans. For each domain i (with $i = 1, \dots, N$), let $\mathcal{J}_i$ denote the set of available service providers. We define the following decision variables:

- $x_{ij} \in \{0, 1\}$: a binary variable that is 1 if provider $j \in \mathcal{J}_i$ is selected for domain i, and 0 otherwise.
- $d_i \geq 0$: the portion of the E2E delay budget allocated to domain i.

The acceptance probability of domain i using provider j when allocated a delay of d_i is given by the function $p_{ij}(d_i)$. Assuming that the decisions made in the domains are statistically independent, the E2E acceptance probability p_{e2e} is modeled as the product of the acceptance probabilities of all domains:

$$p_{\text{e2e}} = \prod_{i=1}^{N} \left(\sum_{j \in \mathcal{J}_i} x_{ij}\, p_{ij}(d_i) \right). \tag{1}$$

The goal is to choose a provider in each domain and allocate the delay budgets $\{d_i\}_{i=1}^{N}$ such that the E2E acceptance probability is maximized, subject to the constraint that the domain-specific delays sum up to the E2E delay budget d_{e2e}. Formally, the optimization problem is formulated as follows:

$$\begin{aligned} \max_{\{x_{ij},\, d_i\}} \quad & \prod_{i=1}^{N} \left(\sum_{j \in \mathcal{J}_i} x_{ij}\, p_{ij}(d_i) \right) \\ \text{s.t.} \quad & \sum_{i=1}^{N} d_i = d_{\text{e2e}}, \\ & d_i \geq 0, \quad \forall\, i = 1, \dots, N, \\ & \sum_{j \in \mathcal{J}_i} x_{ij} = 1, \quad \forall\, i = 1, \dots, N, \\ & x_{ij} \in \{0, 1\}, \quad \forall\, i = 1, \dots, N,\ \forall\, j \in \mathcal{J}_i. \end{aligned} \tag{2}$$

The presence of both integer and continuous variables, coupled with the nonlinear characteristics of the objective function, designates the problem as a canonical MINLP problem.

B. NP-Hardness Analysis

We now demonstrate that the joint optimization problem in (2) is NP-hard by reducing it from the well-known Multiple-Choice Knapsack Problem (MCKP). To this end, consider a simplified version of our problem where the acceptance probability functions $p_{ij}(d_i)$ are defined as threshold functions:

$$p_{ij}(d_i) = \begin{cases} 1, & \text{if } d_i \geq d_{ij}^{\min}, \\ 0, & \text{otherwise}, \end{cases}$$

where $d_{ij}^{\min}$ represents the minimum delay required by provider j in domain i to accept the SLA request. In this simplified version, the acceptance probability of domain i is 1 if and only if the allocated delay d_i meets or exceeds the threshold $d_{ij}^{\min}$ corresponding to the selected provider; otherwise, the acceptance probability is 0. We also assume that $d_{ij}^{\min}$ is known to the orchestrator, which is not the case in our original problem. Moreover, since the domain delay requests must sum to the E2E delay budget d_{e2e}, the thresholds $d_{ij}^{\min}$ effectively serve as the lower bounds for the feasible allocations d_i. Consequently, a necessary condition for a feasible selection of providers is:

$$\sum_{i=1}^{N} d_{i,j(i)}^{\min} \leq d_{\text{e2e}}, \quad \forall \{j(1), j(2), \dots, j(N)\}, \tag{3}$$

where $j(i) \in \mathcal{J}_i$ denotes the provider selected in domain i. This simplified formulation is equivalent to the MCKP. In the MCKP, we are given:

- N disjoint groups, with each group i corresponding to domain i.
- For each group i, a set of items (providers) $j \in \mathcal{J}_i$, where each item is characterized by a weight $d_{ij}^{\min}$ (interpreted as the required delay) and an associated profit v_{ij} (which can be set to 1 for all items, reflecting the binary nature of SLA acceptance).
- A knapsack with capacity d_{e2e} (the E2E delay budget).

The objective in the MCKP is to select exactly one item from each group such that the total weight does not exceed the knapsack capacity d_{e2e} while maximizing the total profit. In our case, achieving a total profit of N (i.e., a profit of 1 from each domain) is equivalent to ensuring that the SLA is accepted across all domains. Since the decision version of the MCKP is NP-complete [13], it follows that the simplified version of our problem is NP-hard. Given that our original problem in (2) generalizes this setting by incorporating more complex acceptance probability functions $p_{ij}(d_i)$, the overall problem is NP-hard as well.

Given the NP-hard nature of the problem, getting an exact solution is computationally intractable for large-scale systems. Hence, we resort to meta-heuristic approaches, leveraging the domain-specific risk models built from historical data to guide the search for near-optimal solutions.

III. Methodology

A. Background

ILS. Iterated Local Search (ILS) [14] is a meta-heuristic approach that enhances local search algorithms by escaping local optima through iterative perturbations and refinements. It is especially useful for combinatorial optimization problems, where the search space is large and complex. ILS operates by first generating an initial solution, either randomly or via a heuristic method, and then refining it through a local search procedure to find a local optimum. Once a local optimum is identified, the algorithm introduces controlled randomness to perturb the solution and push it away from the identified optimum. A predefined acceptance criterion is then applied to determine whether the perturbed solution should replace the current solution. This cycle of local search, perturbation, and acceptance continues until a stopping condition, such as reaching a maximum number of iterations or meeting a convergence criterion, is satisfied. ILS is used in networked cloud resource mapping to address the challenge of optimally partitioning and embedding virtual resources across multiple cloud providers [15]. ILS-based request partitioning has been shown to effectively balance cost and performance, leading to improved virtual network embedding outcomes.

RADE. Real-time Adaptive DEcomposition (RADE) [8] is an advanced SLA decomposition framework that dynamically adjusts decomposition strategies based on real-time feedback of network conditions. Unlike static decomposition approaches, RADE employs online learning to enhance adaptability and accuracy. It utilizes a two-step decomposition approach [6], [7]. First, the orchestrator maintains domain-specific risk models trained on historical SLA acceptance and rejection feedback. Next, the E2E SLA is decomposed into domain-specific SLAs to maximize the overall acceptance probability, using a grid search followed by Sequential Least Squares Programming (SLSQP) algorithm. To adapt to evolving network conditions, these risk models are updated timely via Online Gradient Descent (OGD). A First In First Out (FIFO) memory buffer preserves recent observations, ensuring stable learning while mitigating overfitting caused by transient anomalies. RADE addresses key limitations of static decomposition methods by incorporating real-time adaptation. It also offers resilience against data corruption through its FIFO memory buffer. Experimental results show that RADE consistently outperforms traditional methods in dynamic multi-domain environments, making it a promising solution for adaptive SLA management in modern network architectures.

B. Risk-Aware Iterated Local Search

In this work, we propose RAILS, a risk model-driven meta-heuristic method to solve the joint provider selection and SLA decomposition problem. The optimization problem involves two interconnected sets of decision variables (see Section II-A). On one hand, we have discrete variables that determine which provider is selected in each domain. On the other hand, we have continuous variables that specify how the E2E delay budget is decomposed among the domains to maximize the E2E acceptance probability. These two aspects of the problem are inherently intertwined because the acceptance probability in each domain is computed using risk models that depend on both the selected provider and the assigned delay requests. The risk models, which are constructed from historical data, capture the admission control behavior of each provider as a function of the delay allocation. When a provider is selected for a domain, the corresponding risk model serves as its surrogate, predicting performance for a given delay budget. This means that the evaluation of a potential solution—i.e., a combination of provider selections and delay assignments—cannot be decoupled into two independent problems. A provider might appear attractive under a particular delay allocation in one domain, but the overall E2E performance also depends on the delay allocations and provider choices in other domains. In other words, the discrete and continuous decisions interact nonlinearly through the acceptance probability functions. In RAILS, the framework efficiently explores the complex search space by iteratively refining provider selections with risk models.

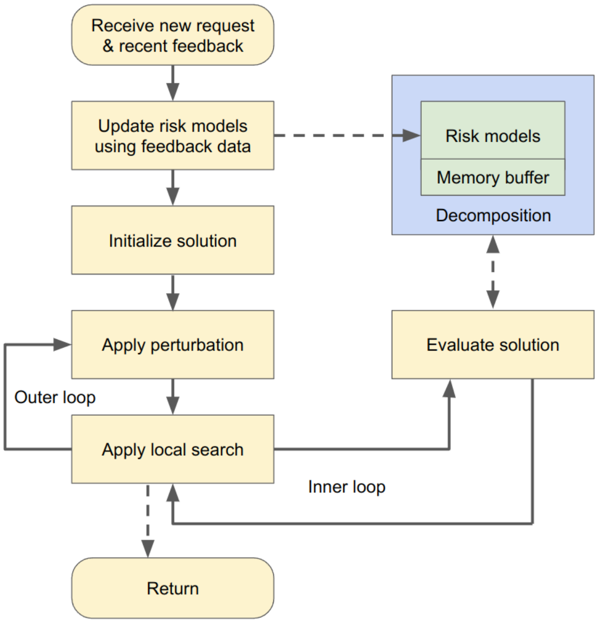

Fig. 1: A single-iteration workflow of RAILS.

Fig. 1 illustrates the workflow of the RAILS algorithm. The process begins with updating the risk models and memory buffer using the latest feedback data. Next, an initial solution for provider selection is generated. The algorithm then enters an iterative refinement phase with perturbation and local search steps. The perturbation step randomly alters the provider selection for each domain with a probability p_μ to possibly escape local optima, while the local search step refines the solution by randomly selecting a domain and exhaustively checking all provider options within the domain to identify the best one based on the risk models. This iterative process continues within an inner and outer loop structure until a predefined stopping condition is met, at which point the best solution identified is returned. RAILS operates within an ILS framework, where the core evaluation mechanism is powered by RADE. Specifically, RAILS performs the repeated perturbation and local search steps, while RADE handles dynamic risk modeling and real-time decomposition. This synergy enables effective handling of the coupled discrete–continuous nature of provider selection and SLA decomposition.

IV. Performance Evaluation

A. Simulation Environment

In our simulation environment, we model the dynamic behavior of each provider's system load and the resulting performance characteristics that affect SLA acceptance. In particular, we capture the temporal variations in load and their impact on the minimum delay that a provider can support, which in turn governs the acceptance probability of an SLA request. Because this subsection focuses on a single-domain provider, we omit the i and j subscripts for clarity.

System Load Modeling. For each provider, the system load is assumed to evolve periodically over time. Let t denote the current time. The system load $\ell(t)$ is modeled using a sinusoidal function as follows:

$$\ell(t) = \ell_{\text{base}} \cdot k + \ell_{\text{base}} \cdot (1-k) \cdot \frac{1+\sin\left(\frac{2\pi t}{T}+\phi\right)}{2}, \tag{4}$$

where ℓ_{base} is a constant representing the baseline load of the provider, $k \in [0,1]$ is a parameter that determines the fraction of the load that is static, T is the period of the sinusoidal fluctuation, and ϕ is the phase shift. This formulation ensures that the system load varies between the minimum load $\ell_{\text{base}} \cdot k$ and the maximum load ℓ_{base}. The parameter k allows for a mixture of a constant baseline load and a dynamic component, thereby enabling the simulation of various operational conditions in real-world network systems.

Minimum Supportable Delay. Given the dynamic system load defined in (4), the minimum delay that a provider can support for an incoming request is assumed to depend on both a fixed latency component and an exponential function of the system load. Specifically, the minimum supportable delay $d^{\min}(t)$ is defined as:

$$d^{\min}(t) = \alpha + \exp(\beta \cdot \ell(t)), \tag{5}$$

where α represents the inherent latency of the system when the load is minimal (i.e., the baseline latency), and β is a parameter that characterizes how sensitive the delay is to changes against system load. This expression reflects that as the system load increases, the provider's capability to handle requests with low delay diminishes, leading to a higher minimum supportable delay. Fig. 2 demonstrates this effect for different parameter values. The exponential relationship captures the non-linear increase in delay requirements as the load intensifies, while α sets the lower bound of latency.

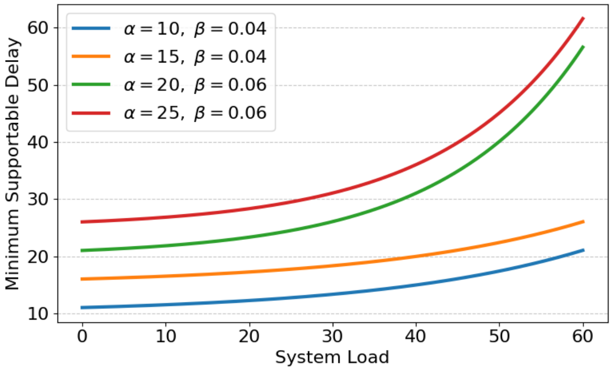

Fig. 2: Minimum supportable delay versus system load.

Acceptance Probability. Based on the minimum supportable delay defined in (5), the acceptance probability of a service level request is defined as a function of the requested delay d. Specifically, if the requested delay is less than the minimum supportable delay $d^{\min}(t)$, the request is rejected. Otherwise, the acceptance probability increases exponentially with the excess delay, saturating as the requested delay becomes much larger than $d^{\min}(t)$ [6]. Formally, the acceptance probability $p(d;t)$ is defined as:

$$p(d;t) = \begin{cases} 0, & \text{if } d < d^{\min}(t), \\ 1-\exp\left(-\lambda\left(d-d^{\min}(t)\right)\right), & \text{if } d \geq d^{\min}(t), \end{cases} \tag{6}$$

where $\lambda > 0$ is a parameter that controls the rate at which the acceptance probability increases as the delay requirement exceeds the minimum supportable delay. The curves with different parameter values are illustrated in Fig. 3.

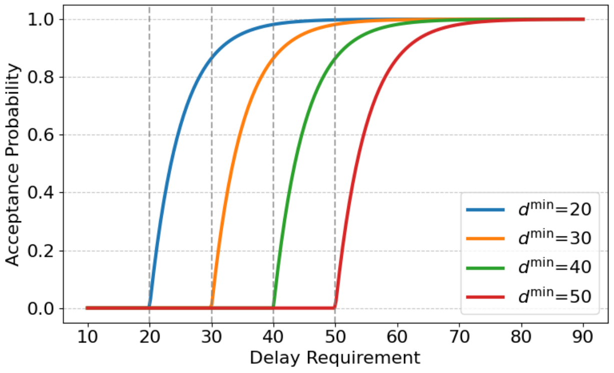

Fig. 3: Piecewise acceptance probability function.

The piecewise function ensures that any request with a delay requirement less than the system's minimum supportable delay is rejected. For requests above this threshold, the probability of acceptance increases rapidly at first and gradually saturates, reflecting the typical behavior of admission control in real-world systems.

B. Evaluation Scenarios

To assess the long-term performance of the proposed RAILS framework, we conduct simulations over 100 discrete time steps within the designed simulation environment. At each time step, a new service request arrives with an E2E delay budget of 100 ms. The RAILS is then applied to select a service provider for each domain and to determine the corresponding delay decomposition that maximizes the E2E acceptance probability according to the risk models built from historical data. Specifically, once the RAILS provides a provider selection and delay decomposition, this assignment is evaluated using the ground-truth acceptance probability models (described in (6)) to compute the actual E2E acceptance probability. This process is repeated at every time step, and the resulting E2E acceptance probabilities are collected to compute an overall average performance metric, namely:

$$\overline{p}_{\text{e2e}} = \frac{1}{T}\sum_{t=1}^{T} p_{\text{e2e}}^{(t)} = \frac{1}{T}\sum_{t=1}^{T}\prod_{i=1}^{N}\left(\sum_{j\in\mathcal{J}_i} x_{ij}^{(t)}\, p_{ij}(d_i^{(t)})\right), \quad (7)$$

where T denote the total number of simulation time steps, and N represents the total number of involved domains.

At each time step, we assume that a set of historical requests and their associated feedback are available from each provider. A historical sample is represented by a pair $(d^{(t)}, a^{(t)})$, where $d^{(t)}$ is the delay request and $a^{(t)} \in \{0, 1\}$ is the binary decision outcome from the admission control. For generating the feedback data, we simulate requests by sampling delay requirements uniformly from the interval $[10\,\text{ms}, 100\,\text{ms}]$. Each request is then processed through the corresponding ground-truth acceptance probability model to obtain its actual acceptance probability, and a coin-flipping process is used to determine whether the request is accepted or not by the admission control. For performance comparison, we consider two baselines:

- **Non-Risk-Aware (NRA).** In this baseline, a provider is selected at random for each domain, and the E2E delay budget is evenly decomposed among the domains as a heuristic guess, without leveraging risk models.
- **Optimal (OPT).** This benchmark is obtained via an exhaustive search with full access to ground-truth acceptance probability models, representing the theoretical upper bound on performance.

By comparing the RAILS approach with these baselines, we aim to demonstrate its effectiveness in achieving higher E2E acceptance probabilities over time, while maintaining reasonable computational time.

C. Experimental Setup

In our experiments, we consider a network slicing scenario involving 3 domains, each comprising 10 service providers. The ground-truth models for each provider are generated using a set of randomly selected parameters to emulate realistic and heterogeneous operational conditions. For each provider, the baseline latency α is drawn uniformly from $[0, 2]$; The parameter λ in (6) is set to 0.2, and k is set to 0.5 in (4); A domain-wise additional latency, randomly chosen from the set $\{0, 10, 20\}$, is added to the baseline latency α to reflect inter-domain behavioral shifts; the load-sensitivity parameter β is sampled uniformly from $[0.04, 0.06]$; the baseline system load ℓ_{base} is drawn uniformly from $[30, 50]$; the period of the sinusoidal load fluctuation is selected as an integer uniformly from the range $[30, 60]$; and the phase shift in the load function is chosen uniformly from the interval $[0, \pi]$.

At each time step, we assume that the number of recent feedback samples available from each provider is proportional to its current system load defined in (4). We use the integer part of the load as the number of samples. For the RAILS framework, the number of iterations is set empirically to the total number of providers across all domains (i.e., $3\times10 = 30$), and the perturbation probability p_μ is 0.8. Results are averaged over 10 independent runs to ensure statistical significance. Each risk model is implemented as a 2-layer monotonic Multi-Layer Perceptron (MLP) with a hidden dimension of 16, similar to that described in [8]. AdamW optimizer is used with a learning rate of 0.01. A memory buffer of size 300 is maintained, and each risk model update involves 10 iterations.

V. Results and Discussion

Fig. 4 presents the average E2E acceptance probability for the three considered approaches. Each colored bar reflects the performance defined in (7) over 10 simulation runs, while the gray bar indicates the inference time. The NRA approach achieves an average acceptance probability of approximately 0.71. The large error bars reveal significant performance fluctuations, suggesting that the absence of strategic decision-making leads to suboptimal performance. In contrast, the proposed RAILS method demonstrates a substantial improvement, achieving an average acceptance probability of around 0.89. The error bars are notably smaller compared to the NRA approach, indicating more stable outcomes. Moreover, we include results for a RAILS variant, denoted RAES, where the ILS search component is replaced with an exhaustive search. RAES's performance reflects RAILS running with an effectively infinite number of iterations. Both RAILS and RAES achieve comparable performance, while RAILS requires only about 31.6% of RAES's inference time, showing its computational efficiency. The OPT approach, which represents the theoretical performance upper bound, achieves an average acceptance probability of about 0.95. The minimal error bars suggest highly consistent performance across all simulations. While RAILS does not fully reach the OPT benchmark, it comes remarkably close, balancing computational efficiency with high SLA acceptance rates.

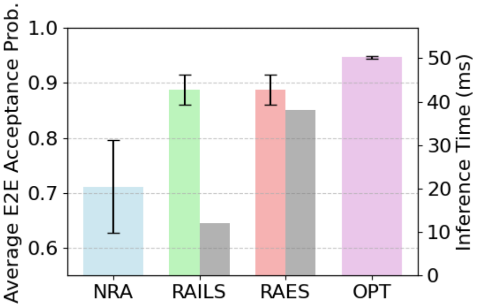

Fig. 4: Average E2E acceptance probabilities over time (colored bars, left axis) and inference time (gray bars, right axis).

TABLE I: Partial trace of a simulation run.

Step	Method	p_{e2e}	d_1, d_2, d_3	Providers
1	NRA	0.00	33.33, 33.33, 33.33	2, 7, 7
	RAILS	0.91	21.35, 35.73, 42.92	5, 5, 3
	OPT	0.94	22.59, 33.63, 43.78	2, 1, 6
5	NRA	0.82	33.33, 33.33, 33.33	9, 2, 3
	RAILS	0.93	21.86, 34.08, 44.06	4, 1, 3
	OPT	0.95	22.78, 33.55, 43.67	2, 1, 2
10	NRA	0.78	33.33, 33.33, 33.33	8, 7, 0
	RAILS	0.91	21.00, 31.00, 48.00	5, 9, 0
	OPT	0.95	23.55, 33.21, 43.24	2, 1, 9
15	NRA	0.77	33.33, 33.33, 33.33	0, 0, 4
	RAILS	0.93	24.00, 30.00, 46.00	5, 8, 3
	OPT	0.95	23.66, 33.22, 43.13	1, 5, 9

Table I provides a partial trace of a simulation run, showcasing the E2E acceptance probabilities, delay decompositions, and the indices of the selected providers across different time steps. The OPT method shows that the optimal provider selections and delay decompositions change frequently over time, reflecting dynamic network conditions. For instance, optimal providers shift from $(2, 1, 6)$ at step 1 to $(1, 5, 9)$ at step 15, highlighting the need for adaptive risk models to maintain high SLA acceptance rates. Furthermore, RAILS achieves near-optimal performance across time steps, even without always selecting the optimal providers.

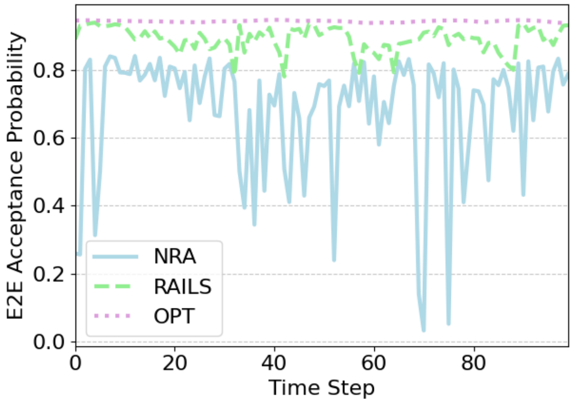

Fig. 5: E2E acceptance probability over time for a single run.

Fig. 5 illustrates the temporal dynamics of E2E acceptance probabilities for a single representative run. The OPT method consistently achieves near-perfect acceptance. RAILS closely tracks the optimal performance, with slight fluctuations, indicating its effectiveness in adapting to dynamic network conditions. In contrast, the NRA method experiences significant variability and frequent drops in acceptance probability, revealing its limitations in predicting and responding to environmental changes.

VI. Conclusion

This paper introduced Risk-Aware Iterated Local Search (RAILS), a meta-heuristic framework driven by risk models, for SLA decomposition and service provider selection in multi-domain networks. We formulated the problem as a Mixed-Integer Nonlinear Programming (MINLP) problem and demonstrated its NP-hardness. RAILS integrates dynamic risk modeling with iterated local search, effectively handling the complex optimization landscape of interdependent decisions. Simulation results showed that RAILS achieves near-optimal performance against the theoretical optimum while maintaining low computational overhead. This highlights RAILS' ability to adapt to dynamic network conditions with high SLA acceptance rates efficiently. Overall, RAILS offers a robust and efficient solution for adaptive network slicing management in modern network systems. Future work will explore the long-term impact of each decision on subsequent ones by formulating the problem as a Markov Decision Process (MDP) and applying Deep Reinforcement Learning (DRL) techniques.

Acknowledgment

This research was partially funded by the HORIZON SNS JU DESIRE6G project (grant no. 101095890) and the Dutch 6G flagship project "Future Network Services".

References

[1] X. Geng, L. M. Contreras, R. Rokui, J. Dong, and I. Bykov, "IETF Network Slice Application in 3GPP 5G End-to-End Network Slice," Internet Engineering Task Force, Internet-Draft draft-ietf-teas-5g-network-slice-application-03, Jun. 2024.

[2] R. Swamy and S. K. M, "5G network slicing," HCL Technologies, Tech. Rep., 2023.

[3] P. Iovanna, M. Svensson, A. Shapin, G. Bottari, F. Ubaldi, F. Ponzini, and M. Puleri, "End-to-end network slicing orchestration," *Ericsson Technology Review*, vol. 2, 2022.

[4] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, "Resource allocation for network slicing in 5G telecommunication networks: A survey of principles and models," *IEEE Network*, vol. 33, no. 6, pp. 172–179, 2019.

[5] J. Wang, J. Liu, J. Li, and N. Kato, "Artificial intelligence-assisted network slicing: Network assurance and service provisioning in 6G," *IEEE Vehicular Technology Magazine*, vol. 18, no. 1, pp. 49–58, 2023.

[6] D. De Vleeschauwer, C. Papagianni, and A. Walid, "Decomposing SLAs for network slicing," *IEEE Communications Letters*, vol. 25, no. 3, pp. 950–954, March 2021.

[7] C. S.-H. Hsu, D. D. Vleeschauwer, and C. Papagianni, "SLA decomposition for network slicing: A deep neural network approach," *IEEE Networking Letters*, pp. 1–1, 2023.

[8] C. S.-H. Hsu, D. De Vleeschauwer, C. Papagianni, and P. Grosso, "Online SLA decomposition: Enabling real-time adaptation to evolving systems," *arXiv preprint arXiv:2408.08968*, 2024.

[9] B. Khodapanah, A. Awada, I. Viering, D. Oehmann, M. Simsek, and G. P. Fettweis, "Fulfillment of service level agreements via slice-aware radio resource management in 5G networks," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, 2018, pp. 1–6.

[10] H. Chergui and C. Verikoukis, "Offline sla-constrained deep learning for 5G networks reliable and dynamic end-to-end slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 350–360, 2020.

[11] A.-C. Maroudis, T. Theodoropoulos, J. Violos, A. Leivadeas, and K. Tserpes, "Leveraging graph neural networks for sla violation prediction in cloud computing," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 605–620, 2024.

[12] M. Iannelli, M. R. Rahman, N. Choi, and L. Wang, "Applying machine learning to end-to-end slice SLA decomposition," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 92–99.

[13] H. Kellerer, U. Pferschy, and D. Pisinger, *The Multiple-Choice Knapsack Problem*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 317–347.

[14] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated Local Search*. Boston, MA: Springer US, 2003, pp. 320–353.

[15] A. Leivadeas, C. Papagianni, and S. Papavassiliou, "Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1077–1086, 2013.