

Joint Network Slicing, Routing, and In- Network Computing for Energy-Efficient 6G

ZEINAB SASAN*, MASOUD SHOKRNEZHAD†, SIAVASH KHORSANDI*, AND TARIK TALEB†‡

* AMIRKABIR UNIVERSITY OF TECHNOLOGY, TEHRAN, IRAN; {Z.SASAN, KHORSANDI}@AUT.AC.IR

† OULU UNIVERSITY, OULU, FINLAND; {MASOUD.SHOKRNEZHAD, TARIK.TALEB}@OULU.FI

‡ RUHR UNIVERSITY BOCHUM, BOCHUM, GERMANY; TARIK.TALEB@RUB.DE

Table of Content

Introduction

System Model and Problem Formulation

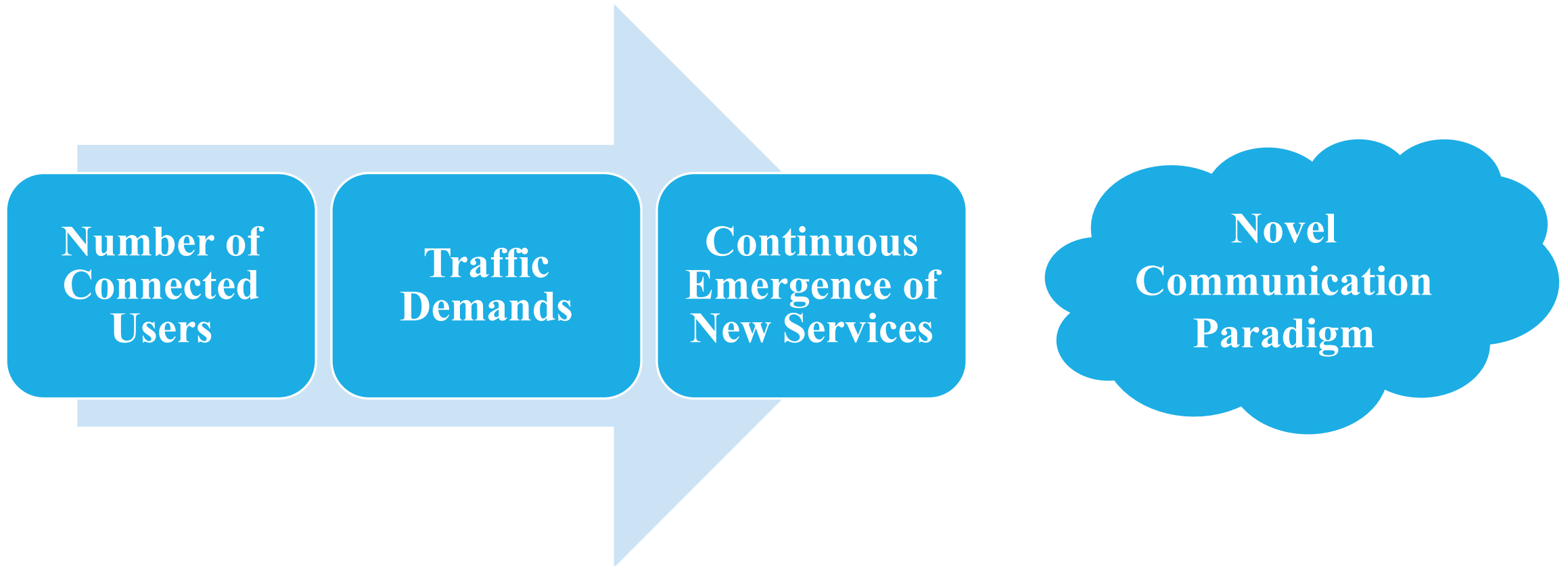
Heuristic Algorithm

Implementation

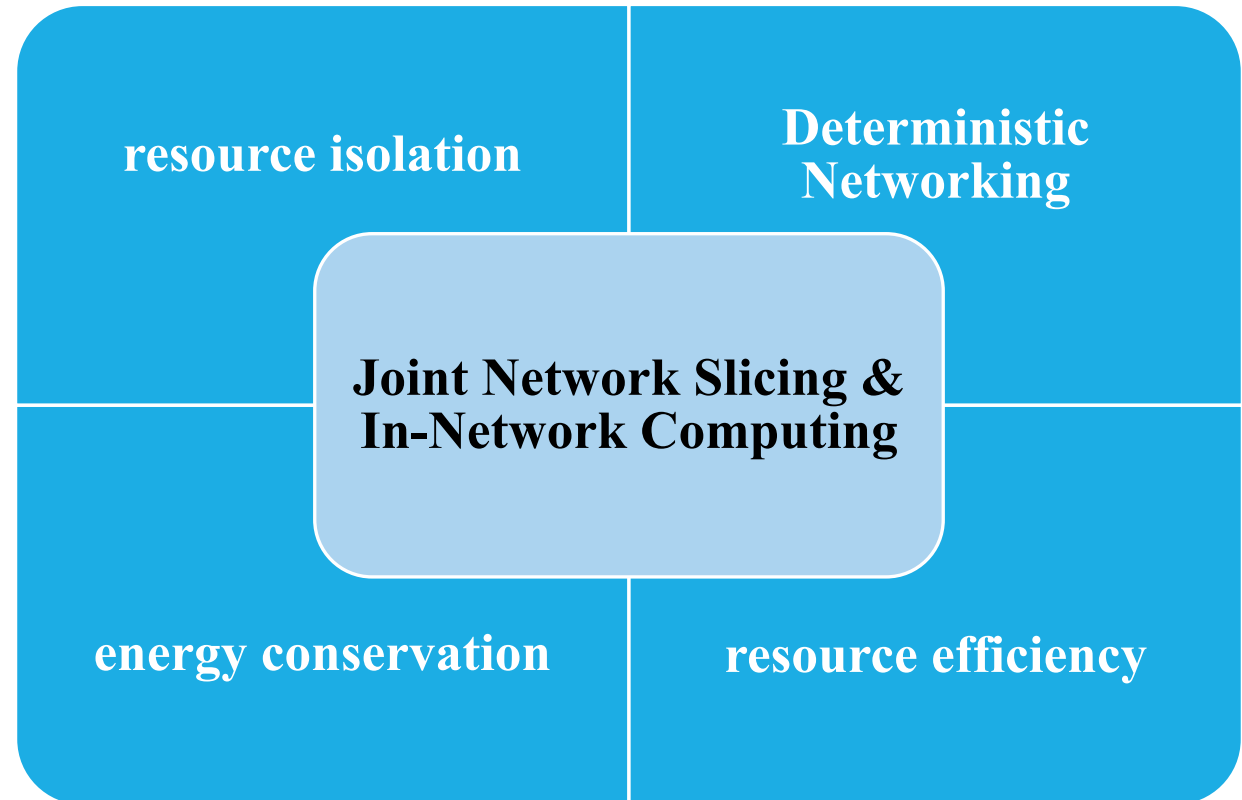
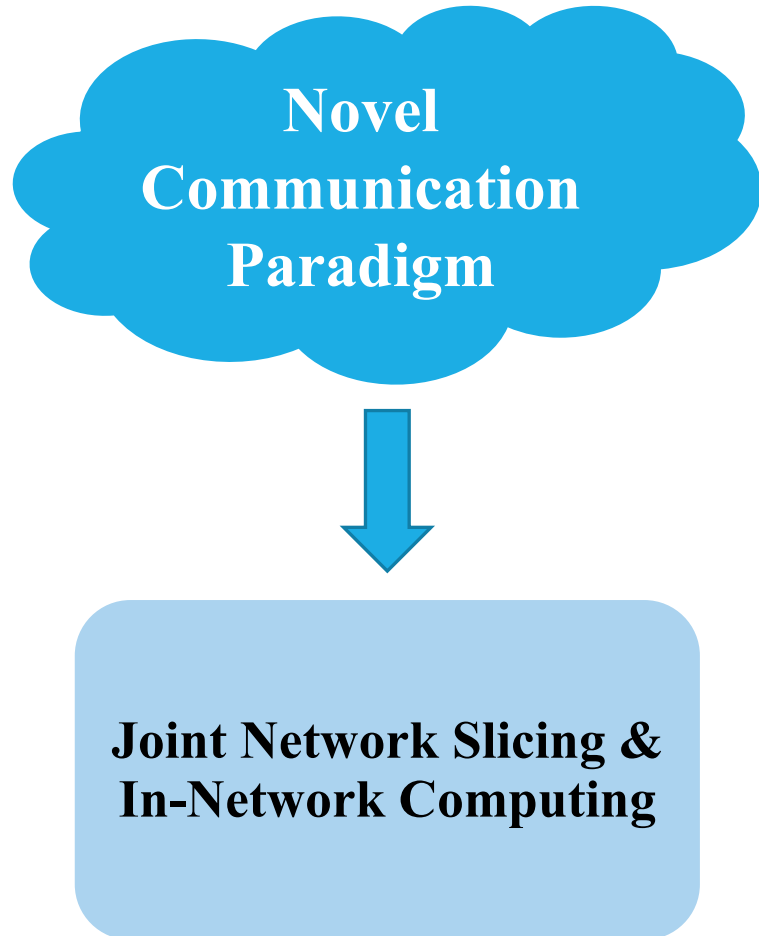
Performance Evaluation

Conclusion

Introduction



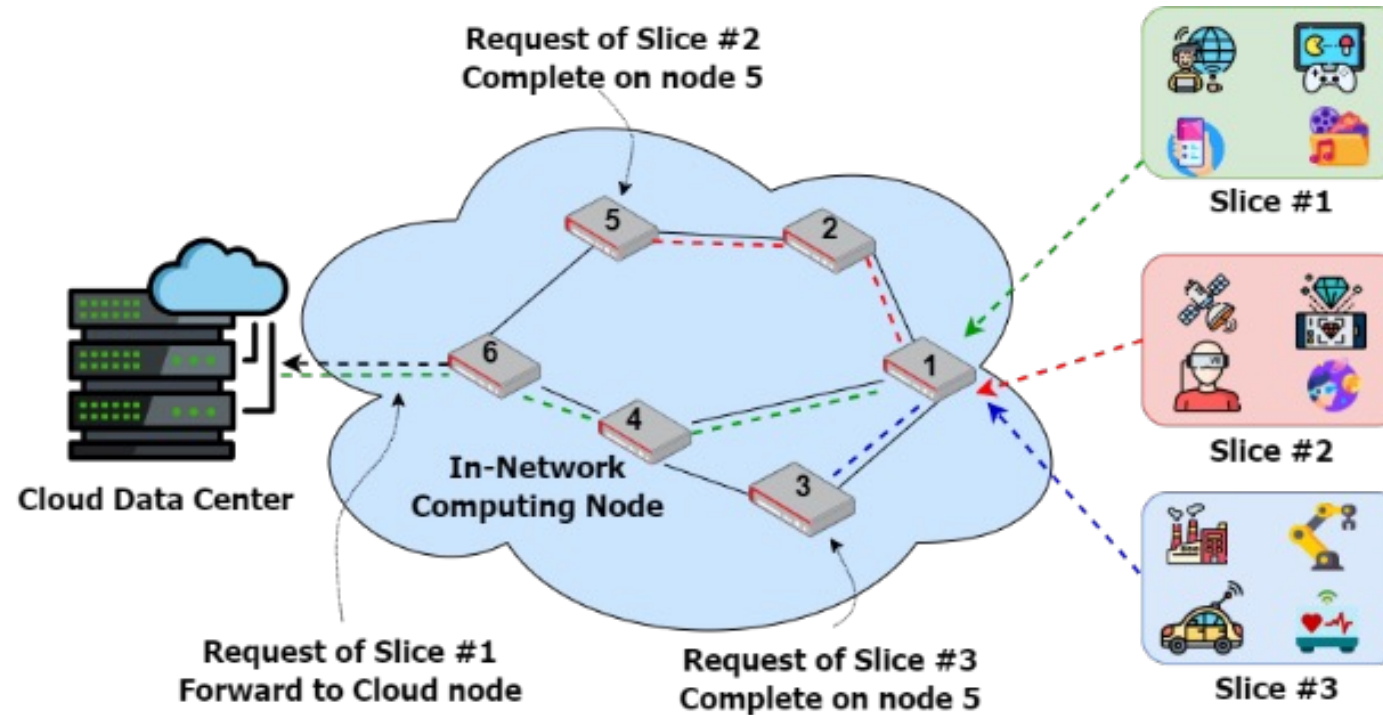
Introduction



Contributions

- Defining the Mixed-Integer Linear Programming (MILP) formulation
 - joint problem of network slicing, routing, and in-network computing
 - maximizing the number of accepted users while minimizing energy consumption
 - End-to-End (E2E) capacity and quality of service constraints
- Proposing a Water-Filling-based Joint Slicing, Routing, and In-Network Computing (WF-JSRIN) heuristic algorithm
- Assessing the efficiency of WF-JSRIN with:
 - Random allocation method
 - Opt-IN (which leverages in-network computation)
 - Opt-C (which relies solely on cloud node resources)

System Model



**The conceptual overview of joint network slicing, routing,
and in-network computing**

Problem Constraints

$$\sum_{p \in \mathbb{P}} x_{u_m}^p \leq 1 \quad \forall m, u_m \in \mathbb{M}, \mathbb{U}_m \quad (\text{C1}) \quad \text{Path Selection \& Admission Control}$$

$$\sum_{v \in \mathbb{V}_p} w_{u_m}^{p,v} = r_{u_m} \cdot x_{u_m}^p \quad \forall m, u_m, p \in \mathbb{M}, \mathbb{U}_m, \mathbb{P} \quad (\text{C2}) \quad \text{Data Rate Satisfaction for } u_m \text{ on Selected Path } p$$

$$w_{u_m}^{p,v} \leq \mathcal{M} \cdot z_{u_m}^{p,e} \quad \forall m, u_m, p, v, e \in \mathbb{M}, \mathbb{U}_m, \mathbb{P}, \mathbb{V}_p, \mathbb{E}_{p,v} \quad (\text{C3}) \quad \text{Identify the Specific Links that Each Request Occupies Along Selected Path } p$$

$$\sum_{m \in \mathbb{M}} \sum_{u_m \in \mathbb{U}_m} \sum_{p \in \mathbb{P}} w_{u_m}^{p,v} \leq \mathcal{M} \cdot y_v \quad \forall v \in \mathbb{V} \quad (\text{C4}) \quad \text{Turn on/off Indicator of Active/Idle Nodes}$$

$$\sum_{u_m \in \mathbb{U}_m} \sum_{p \in \mathbb{P}} w_{u_m}^{p,v} \leq \lambda_m \cdot c_v \quad \forall m, v \in \mathbb{M}, \mathbb{V} \quad (\text{C5}) \quad \text{Enforcing Isolation Between slices on Links \& Nodes capacities}$$

$$\sum_{u_m \in \mathbb{U}_m} \sum_{p \in \mathbb{P}} z_{u_m}^{p,e} \cdot r_{u_m} \leq \lambda_m \cdot c_e, \quad \forall m, e \in \mathbb{M}, \mathbb{E} \quad (\text{C6})$$

$$\lambda_m \geq \epsilon_m, \quad \forall m \in \mathbb{M} \quad (\text{C7}) \quad \text{Minimum Allocation for Each Slice}$$

$$\sum_{m \in \mathbb{M}} \lambda_m = 1, \quad 0 \leq \lambda_m \leq 1 \quad (\text{C8}) \quad \text{Aggregate of Allocations Among All Slices Equates to 1}$$

$$\sum_{v \in \mathbb{V}_p} d_v \cdot w_{u_m}^{p,v} + \sum_{e \in \mathbb{E}_p} d_e \cdot z_{u_m}^{p,e} \cdot r_{u_m} \leq d_{u_m} \quad \forall u_m, p \in \mathbb{U}_m, \mathbb{P} \quad (\text{C9}) \quad \text{E2E Delay (Computation and Networking Delays) not Exceed the Delay Requirement for the Request}$$

Objective Function

$$V_{ec} = \sum_{m \in \mathbb{M}} \sum_{u_m \in \mathbb{U}_m} \sum_{p \in \mathbb{P}} \sum_{v \in \mathbb{V}_p} w_{u_m}^{p,v} \cdot \delta_v$$

(1) Cost of Computations on Nodes

$$P_{ec} = \sum_{v \in \mathbb{V}} y_v \cdot \theta_v$$

(2) Cost of Nodes Activation

$$E_{ec} = \sum_{m \in \mathbb{M}} \sum_{u_m \in \mathbb{U}_m} \sum_{p \in \mathbb{P}} \sum_{e \in \mathbb{E}_p} z_{u_m}^{p,e} \cdot r_{u_m} \cdot \gamma_e$$

(3) Cost of Data Transmission over Links

$$A = \sum_{m \in \mathbb{M}} \sum_{u_m \in \mathbb{U}_m} \sum_{p \in \mathbb{P}} x_{u_m}^p$$

(4) Count of Accepted Users (A)

$$B = V_{ec} + E_{ec} + P_{ec}$$

(5) Cost of Energy Consumption (B)

$$\max_{\lambda, x, y, w, z} \alpha A - B \text{ s.t. } C1 - C9$$

(6) Maximize the count of accepted users (A) while concurrently Minimizing the cost of energy consumption (B)

Heuristic Algorithm

- Initialization:
 - Initialize decision variables (λ, x, y, z, w).
 - Compute total data rate (d_t) and slice-specific data rates (d_m).
 - Adjust λ values based on fairness parameters.
- Capacity Initialization:
 - Initialize node and link capacities for each slice.
- For Each User:
 - For each user, repeat the following steps:
 - Initialize user's capacity requirement and path costs.
 - Path Iteration:
 - Iterate through potential paths:
 - Check if resources are available and allocate if possible.
 - Reserve link capacity if computational requirement is met.
 - Discard allocation if resources are insufficient or delay threshold is exceeded.
- Path Selection:
 - Choose path with minimum cost.

Algorithm 1: WF-JSRIN

Input: $G, M, \epsilon_m \forall m \in M$

- 1 Initialize λ, x, y, z, w
- 2 d_t : total data rate of all slices
- 3 **for each** m **in** M **do**
- 4 $d_m \leftarrow$ total data rate of the slice m
- 5 $\lambda_m \leftarrow \epsilon_m + (1 - \epsilon_m * |M|)d_m/d_t$
- 6 **for each** m **in** M **do**
- 7 $c_{v,m} \leftarrow$ remaining capacity of nodes for slice m
- 8 $c_{e,m} \leftarrow$ remaining capacity of links for slice m
- 9 **for each** u_m **in** U_m **do**
- 10 $\phi_{u_m} \leftarrow$ the data rate requirement of u_m
- 11 $\mu_{u_m,p} \leftarrow 0$ for each p in \mathbb{P}
- 12 $\varphi_{u_m,p} \leftarrow 0$ for each p in \mathbb{P}
- 13 **for each** p **in** user's paths **do**
- 14 reset ϕ_{u_m}
- 15 **for each** v **in** p **do**
- 16 **if** $\phi_{u_m} \neq 0$ & $c_{v,m} \neq 0$ **then**
- 17 $y_v \leftarrow 1$
- 18 $w_{u_m}^{p,v} \leftarrow \min\{\phi_{u_m}, c_{v,m}\}$
- 19 $\Omega = \phi_{u_m} - c_{v,m}$
- 20 $\phi_{u_m} \leftarrow \max\{0, \Omega\}$
- 21 $c_{v,m} \leftarrow \max\{0, -\Omega\}$
- 22 Update $\varphi_{u_m,p}$ & $\mu_{u_m,p}$
- 23 **if** $\phi_{u_m} = 0$ **then**
- 24 **for each** e **in** $\mathbb{E}_{p,v}$ **do**
- 25 **if** $c_{e,m} \geq r_{u_m}$ **then**
- 26 $z_{u_m}^{p,e} \leftarrow 1$
- 27 $c_{e,m} \leftarrow c_{e,m} - r_{u_m}$
- 28 Update $\varphi_{u_m,p}$ & $\mu_{u_m,p}$
- 29 **else**
- 30 reset allocations and updates
- 31 **if** $\varphi_{u_m,p} > d_{u_m}$ **then**
- 32 reset allocations and updates
- 33 **if** $\phi_{u_m} \neq 0$ **then**
- 34 reset allocations and updates
- 35 $p' \leftarrow \operatorname{argmax}_{p \in \mathbb{P}} \mu_{u_m,p}$
- 36 $x_{u_m}^{p'} \leftarrow 1$
- 37 **return** λ, x, y, z, w

Implementation

- Execution Environment:
 - 8 processing cores
 - 16 GB of memory
 - 64-bit operating system
- Comparative Analysis between:
 - Water-Filling-based Joint Slicing, Routing, and In-Network Computing (WF-JSRIN)
 - Random- Based Joint Slicing, Routing, and In-Network Computing (R-JSRIN)
 - Opt-IN (leverages in-network computation)
 - Opt-C (relies solely on cloud node resources)

SIMULATION PARAMETERS

Parameters	Values
number of nodes in graph G	12
cloud node capacity	$\sim U[200, 300]$
nodes capacity (c_v)	$\sim U[15, 50]$
links capacity (c_e)	$\sim U[100, 200]$
links & nodes delay (d_e, d_v)	$\sim U[1, 3]$
links & nodes cost (θ_e, δ_v , and θ_v)	$\sim U[1, 5]$
data rate of users (r_{u_m})	$\sim U[10, 20]$
delay of users (d_{u_m})	$\sim U[500, 1000]$

Performance Evaluation

- Execution times of **WF-JSRIN** and **R-JSRIN** methods consistently less than one second, even with increasing user count.
- Opt-C method, relying solely on cloud node, shows reduced execution time compared to Opt-IN.
- Opt-IN method exhibits **exponential increase** in execution time with growing user count due to complex resource distribution decisions.

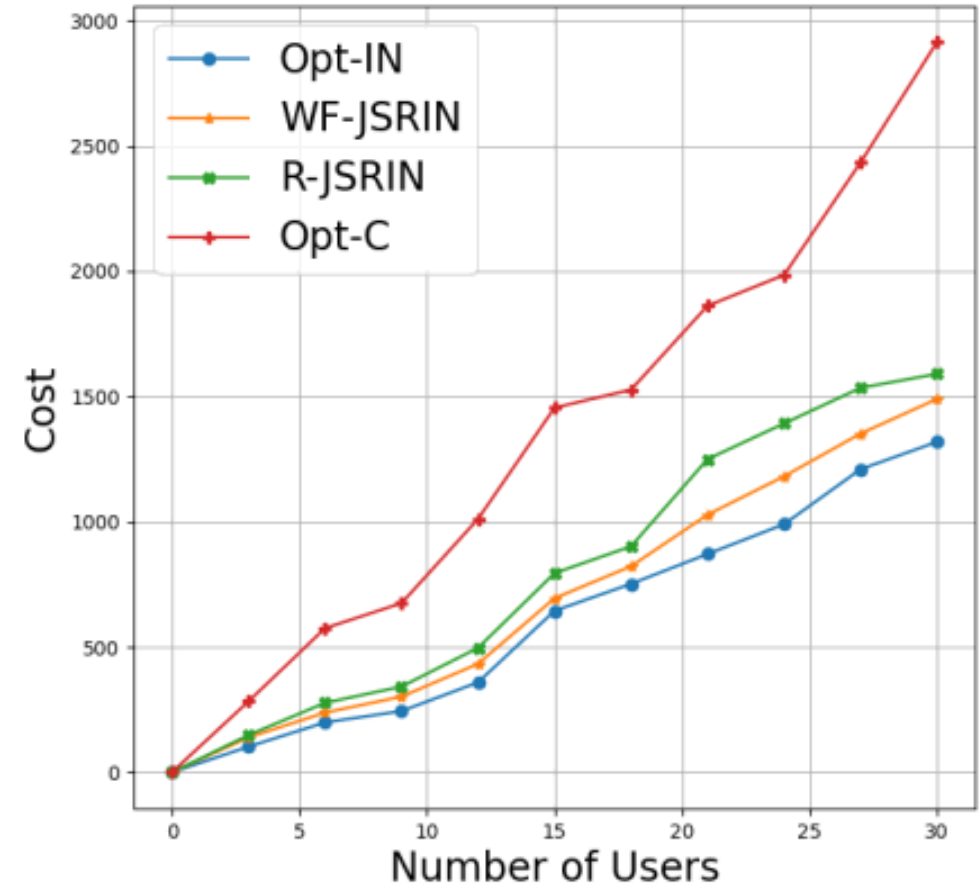
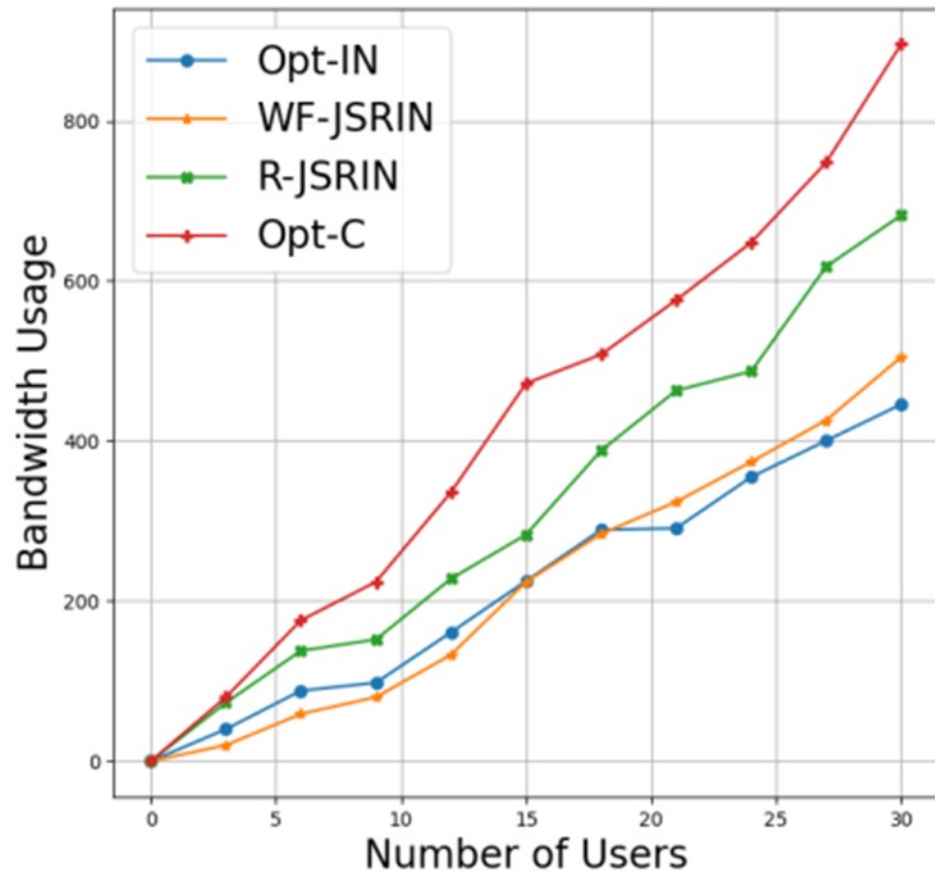
SOLVING TIME VS. THE NUMBER OF USERS (SECOND)

Users Num	Opt-IN	Opt-C	WF-JSRIN	R-JSRIN
3	0.528	0.0305	0.015	0.015
6	0.530	0.060	0.018	0.030
9	4.015	0.120	0.034	0.038
12	101.874	0.176	0.054	0.062
15	129.460	0.263	0.089	0.087
18	405.497	0.383	0.124	0.132
21	3474.588	0.574	0.183	0.180
24	6543.679	2.287	0.234	0.217
27	9612.770	2.856	0.245	0.228
30	12681.8615	3.337	0.361	0.352

Empirical findings suggest that optimal methodologies incur significant time overhead in medium and large-scale networks.

Preference should be given to heuristic algorithms for time efficiency in such scenarios.

Performance Evaluation

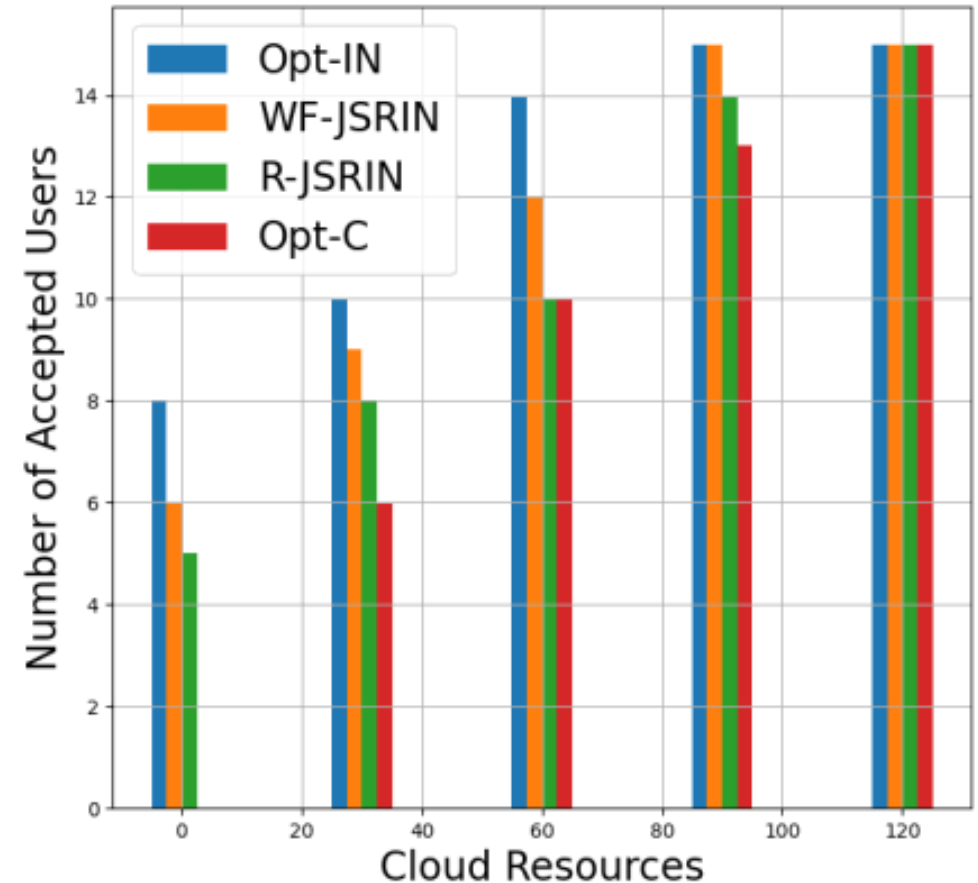


Performance Evaluation

- Opt-C consistently incurs higher costs and greater bandwidth usage compared to other algorithms due to:
 - Requests traversing more links to reach the cloud, leading to increased energy consumption and resource usage.
 - Processing at the cloud node being costlier than at intermediary nodes within the network.
- R-JSRIN consumes more bandwidth and incurs higher costs than Opt-IN and WF-JSRIN because:
 - It may select more distant nodes for request distribution, increasing bandwidth usage.
- WF-JSRIN utilizes less bandwidth than Opt-IN by initiating computation distribution from closer nodes, resulting in fewer traversed links for small numbers of users.
- As the number of users increases, WF-JSRIN navigates more links, causing its bandwidth usage to surpass that of Opt-IN.
- WF-JSRIN achieves similar results to Opt-IN, aiming to minimize total energy consumption and maximize the number of accepted users, demonstrating near-optimal efficiency even for large numbers of users.

Performance Evaluation

- Augmenting cloud node resources leads to a proportional increase in users accepted by the Opt-C approach.
- Remarkably, even with limited cloud resources, all approaches leveraging in-network computation outperform Opt-C.
- WF-JSRIN closely resembles Opt-IN's performance, demonstrating its effectiveness even in scenarios with constrained cloud resources.



Performance Evaluation

- Impact of varying link capacities on user acceptance across different algorithms:
 - In situations with constrained link capacity, in-network computation excels at accommodating more users by avoiding transmission of requests to the cloud node, optimizing resource utilization.
 - With expanded link capacity, Opt-IN and Opt-C achieve equitable distribution of accepted users.
 - R-JSRIN algorithm may select a distant node for request computation, requiring traversal through numerous links. In cases of inadequate link capacity, R-JSRIN lags behind WF-JSRIN.

Conclusion

- Results highlight WF-JSRIN's ability to provide highly efficient near-optimal solutions
- WF-JSRIN also demonstrates remarkably shorter execution times

- Future Works:
 - Address the dynamic nature of the problem by integrating machine learning techniques
 - Integration of radio access resource allocation and integrated Access and Backhaul (IAB)-enabled E2E resource allocation
 - Integration of slicing for Quantum Networks (QNs) with in-network computation