Algorithm for optimized service orchestration

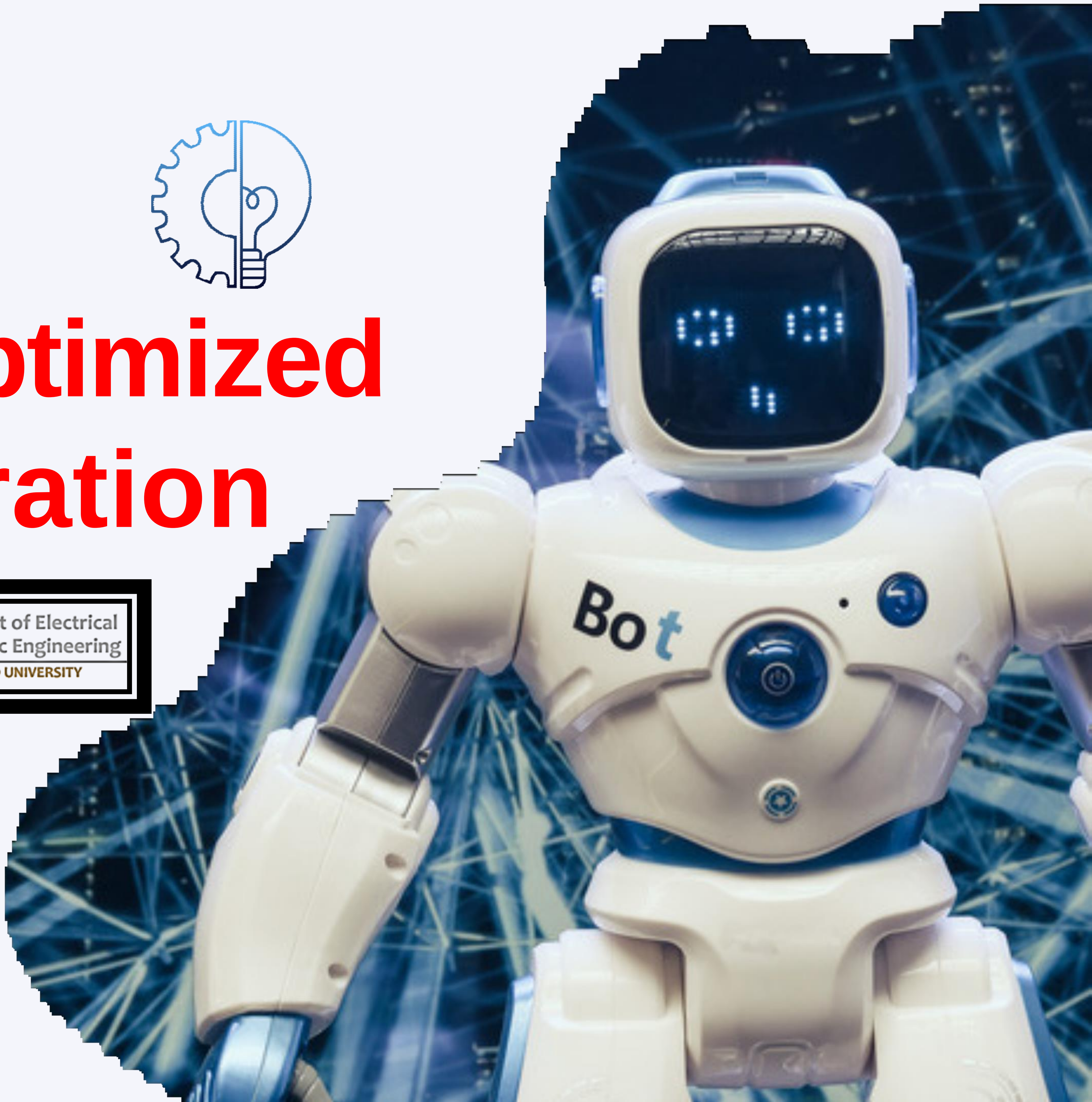
Kyambogo University
Knowledge and skills for service

Department of Electrical & Electronic Engineering
KYAMBOGO UNIVERSITY

ICCECIP conference 2023

BORSOS DONIZ
SUZAN BABIRYE

Dr. ABEL KAMAGARA



Background

Service orchestration is the process involved in designing, creating, and delivering end-to-end services.

- Traditionally, these processes were handled by domain-specific, siloed operational support systems and tools built for static environments.
- With the introduction of Network Functions Virtualization (NFV) and Software-Defined Networking (SDN), service orchestration must take a new approach in serving the needs of today’s more dynamic and complex service provider environments.

Problem Statement

The primary challenge in service orchestration is to determine the optimal combination of services that can be used to achieve a specific business objective while minimizing costs.

This problem is complex, especially in large-scale systems with many possible combinations of services and resources.

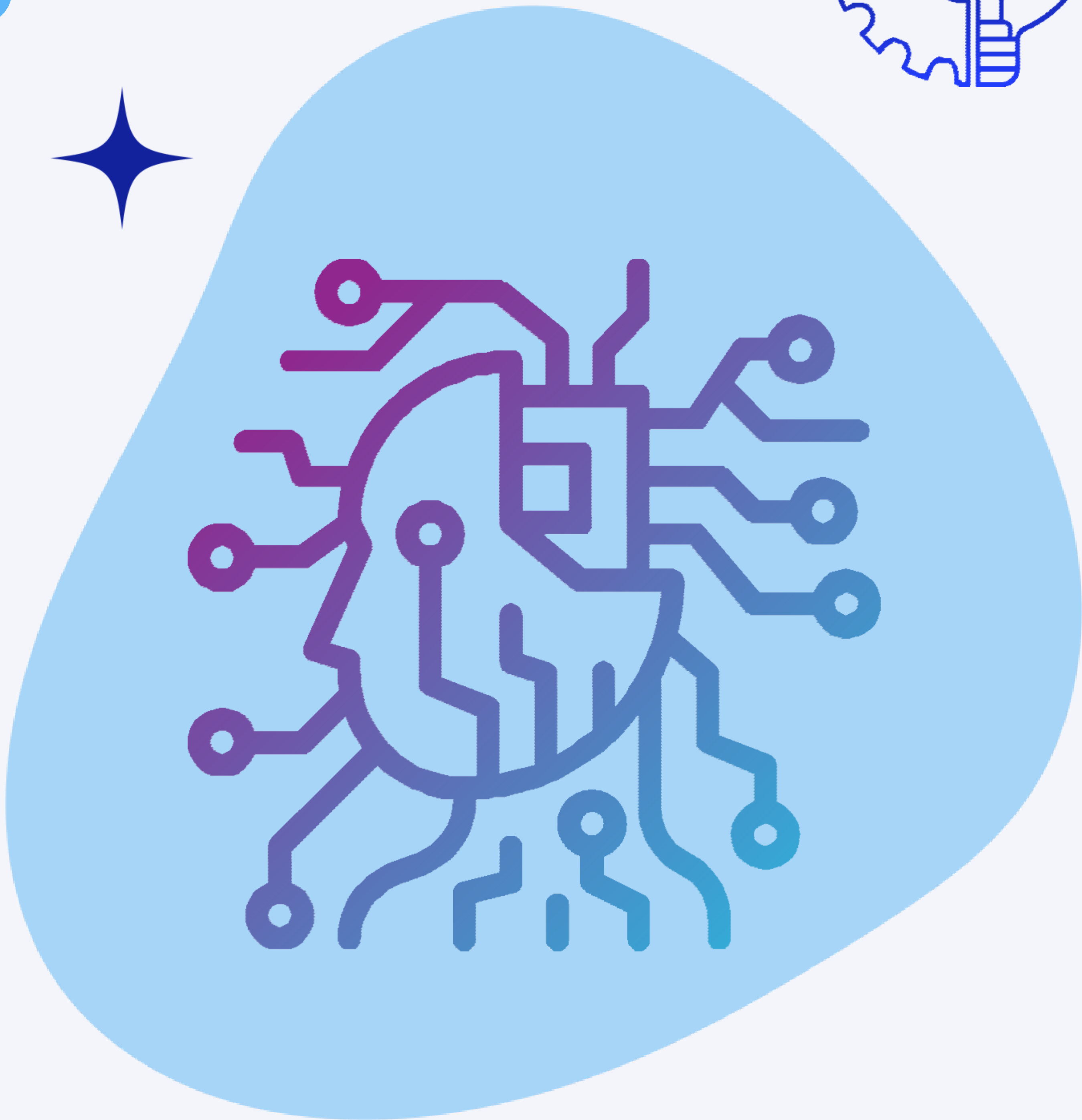
Objectives

General Objective

Develop a systematic approach for service selection, resource allocation, and service composition.

Specific Objective

Design the algorithm with the concept of ILP, simuate it in MATLAB and evaluate its performance in terms of cost and service delivery.



Justification

Increased efficiency

By automating the process of selecting the optimal set of services for a given request, the algorithm will enable faster delivery and reduce the need for manual intervention.

Cost reduction

By selecting the most cost-effective combination of services, the algorithm will help organizations save money and increase profitability

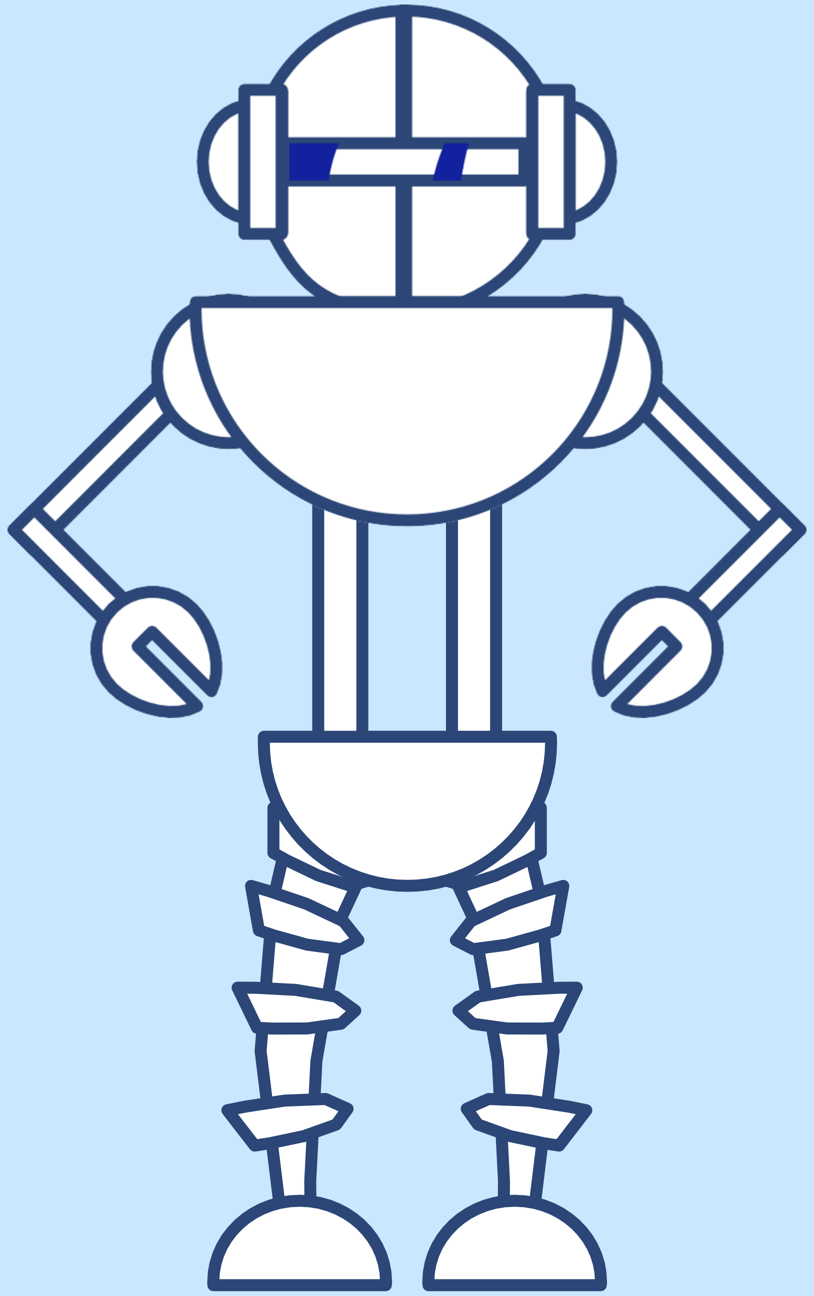
Improved quality of service

By selecting the best combination of services for a given request, the algorithm can improve the overall quality of service.

Significance

Improved SLAs

Rapid DevOps



Operational piece of mind

Greater efficiency & cost savings

Scope

- The algorithm will be designed to minimize the total cost of service orchestration while satisfying the latency requirements of the application.
- The algorithm will be evaluated using simulation experiments to demonstrate its effectiveness in achieving optimal service orchestration.
- The study will focus on a theoretical model of the problem and will not include the implementation of the algorithm in a real-world system.

Literature Review

ILP

ILP is used to determine the optimal sequence of service execution and resource allocation to minimize cost and maximize performance.

OpenAPI

OpenAPI enables service orchestration by providing a standard way to interface with different services and applications.

Multi-objective optimization

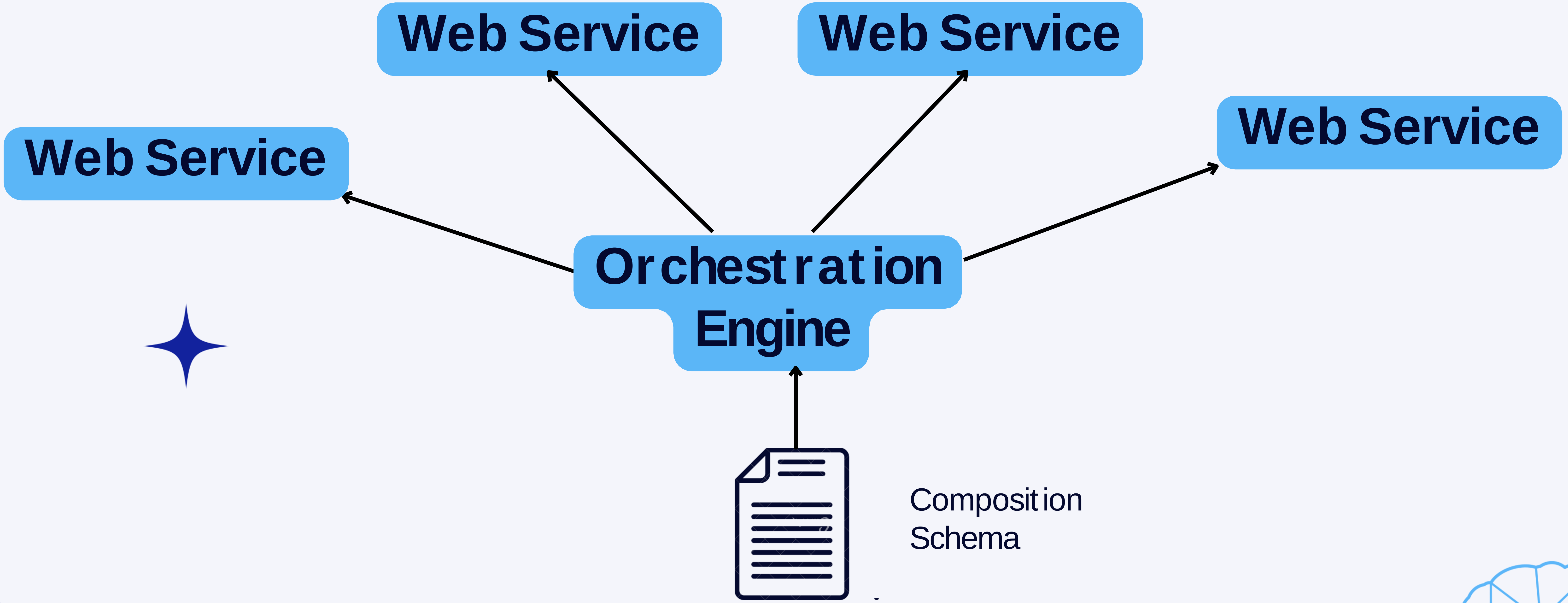
In service orchestration, multi-objective optimization is used to balance performance and cost objectives while satisfying QoS constraints.

Methodology

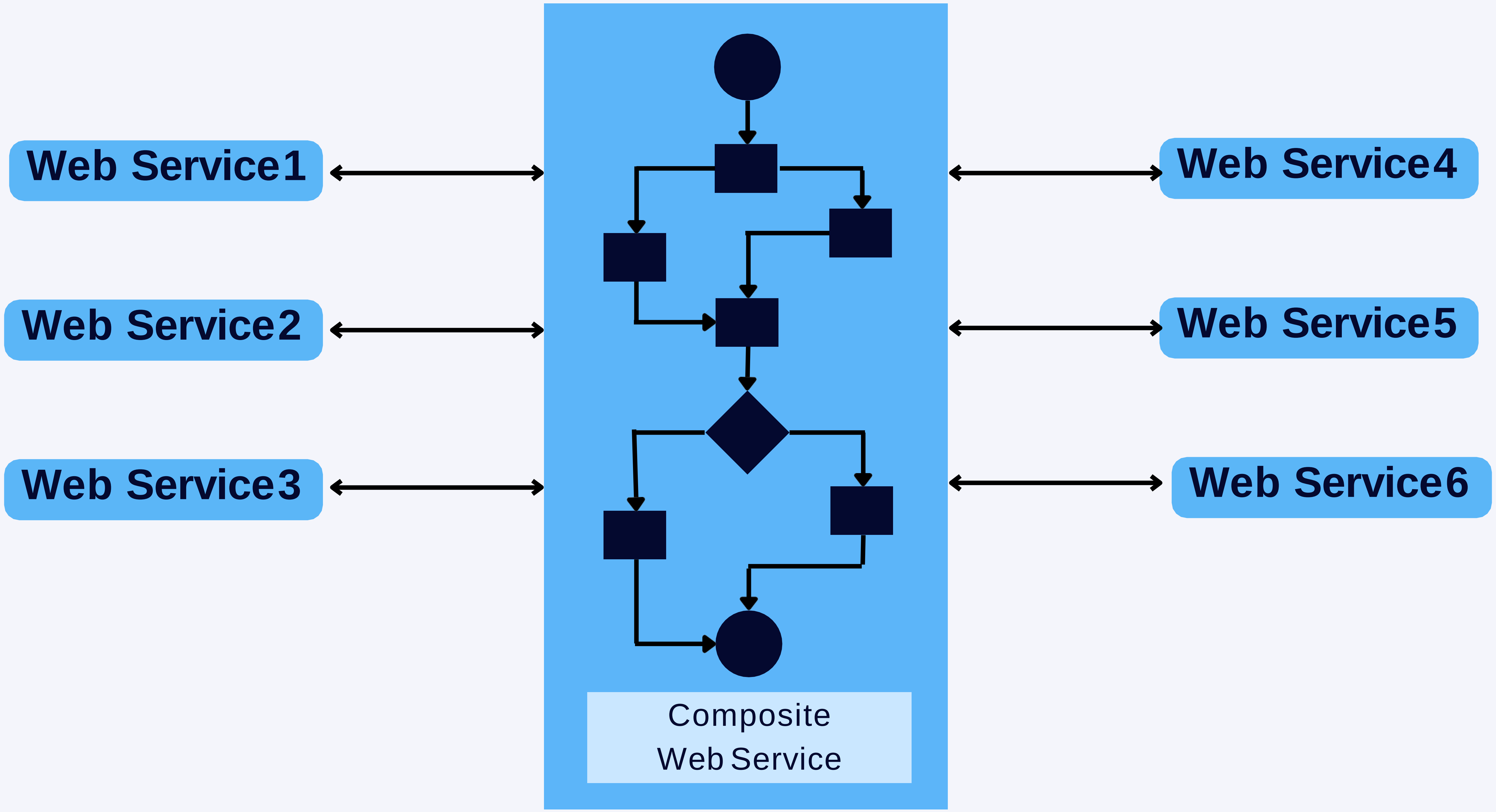
- This paper presents a software-defined architecture for IoT service orchestration.

- The theory is based on the concept of integer linear programming (ILP) to formulate the optimization problem and solve it using MATLAB’s optimization toolbox.

Block representation of Service Ochestration



Composition Schema



Simulation design

- Defining the problem
- Modelling the problem
- Formulating the ILPproblem

Objective Function:

$$minimize\ C(A) = \sum(cij * xij) + \sum(dj * yj) \quad (1)$$

Where,

$C(A)$ is the total cost of the services used, cij is the cost of using service j at network element I, xij is a binary variable that is 1 if service j is used at network element i, and 0 otherwise, dj is the cost of deploying service j, yj is a binary variable that is 1 if service j is deployed, and 0 otherwise.

Formulating the ILP problem (...cont)

Subject to:

Service Constraints:

$\Sigma(xij) = 1$ for each i, where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ (2)

This constraint ensures that each network element uses only one service.

Capacity Constraints:

$\Sigma(xij * bwj) \leq 1$ for each i, where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ (3)

This constraint ensures that the bandwidth used by each service does not exceed the capacity of the network element.

Deployment Constraints:

$xij \leq yj$ for each j, $j \in \{1, \ldots, m\}$ (4)

This constraint ensures that a service can only be used if it is deployed.

Resource Constraints:

$\Sigma(yj) \leq k$ (5)

This constraint limits the number of services that can be deployed to k.

Simulation Code Execution Order

1. Define problem variables: number of network elements, number of services, costs of using services at each network element, and costs of deploying services.
2. Define the objective function to minimize the total cost of the services used.
3. Define the constraints that ensure that each service is deployed at only one network element and that each network element deploys only one service.
4. Define the binary decision variables for each combination of network element and service.
5. Use the YALMIP toolbox to create the MILP problem.

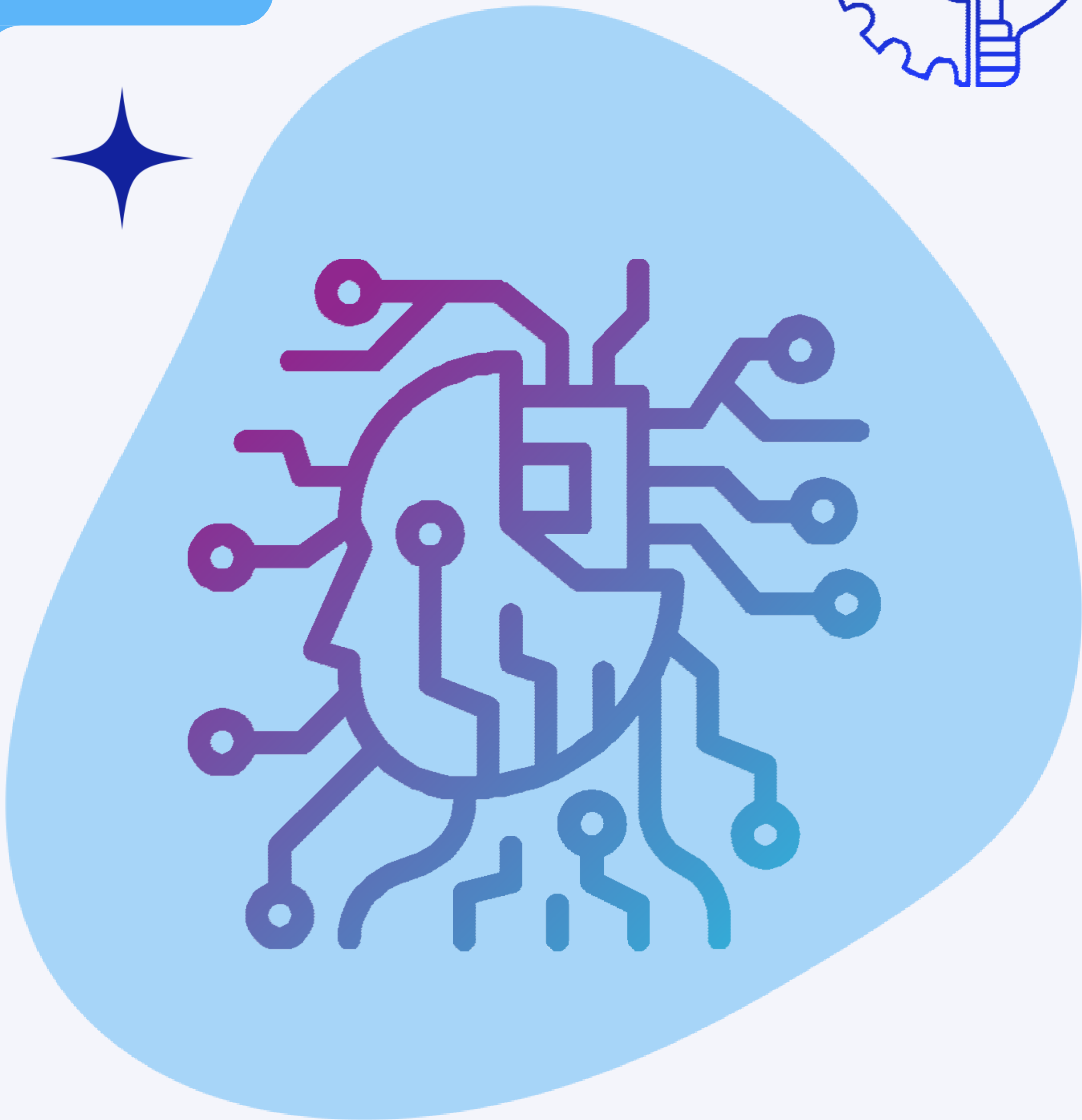
Simulation Code Execution Order (cont...)

6. Use the intlinprog function to solve the MILP problem.

7. Compute the cost and execution time for the solution.

8. Repeat steps 1-7 for different numbers of network elements to generate plots of cost and execution time vs number of network elements.

9. Output the results and plot the comparison graphs



Results & Discussion

Fig 1: Simulation Code snippet

Editor - C:\Users\Frankcistems\Documents\MATLAB\altsolution.m

solve_ilp_problem.m | altsolution.m | ilpPlot.m | +

```
n = 3; % number of micro-services
fc = randi([1 10],n,n); % cost of micro-service
ft = randi([10 100],n,n); % delay of micro-service
fce = randi([1 5],n,1); % cost of contextual service (transpose to make it a column vector)
fte = randi([1 10],1,n); % delay of contextual service

g = 500; % latency constraint

% Define the binary decision variables
x = binvar(n,n,'full'); % x(i,j) is the decision variable for selecting the j-th provider for the i-th micro-service
y = binvar(n,1,'full'); % y(k) is the decision variable for selecting the k-th contextual service

% Define the objective function
obj = sum(sum(fc.*x)) + sum(fce.*y);

% Define the constraints
constr = [sum(x,2) == 1; % each micro-service can only be provided by one provider
    sum(y) >= 1; % only one contextual service can be selected
    sum(ft.*x,2).' + fte*y <= g]; % the total latency of the application must be less than the constraint

% Solve the ILP problem
options = sdpsettings('solver','intlinprog');
tic;
sol = optimize(constr,obj,options);
execution_time = toc;

% Evaluate solution quality
if sol.problem == 0
    cost = value(obj);
else
    cost = Inf;
end
```

Results & Discussion

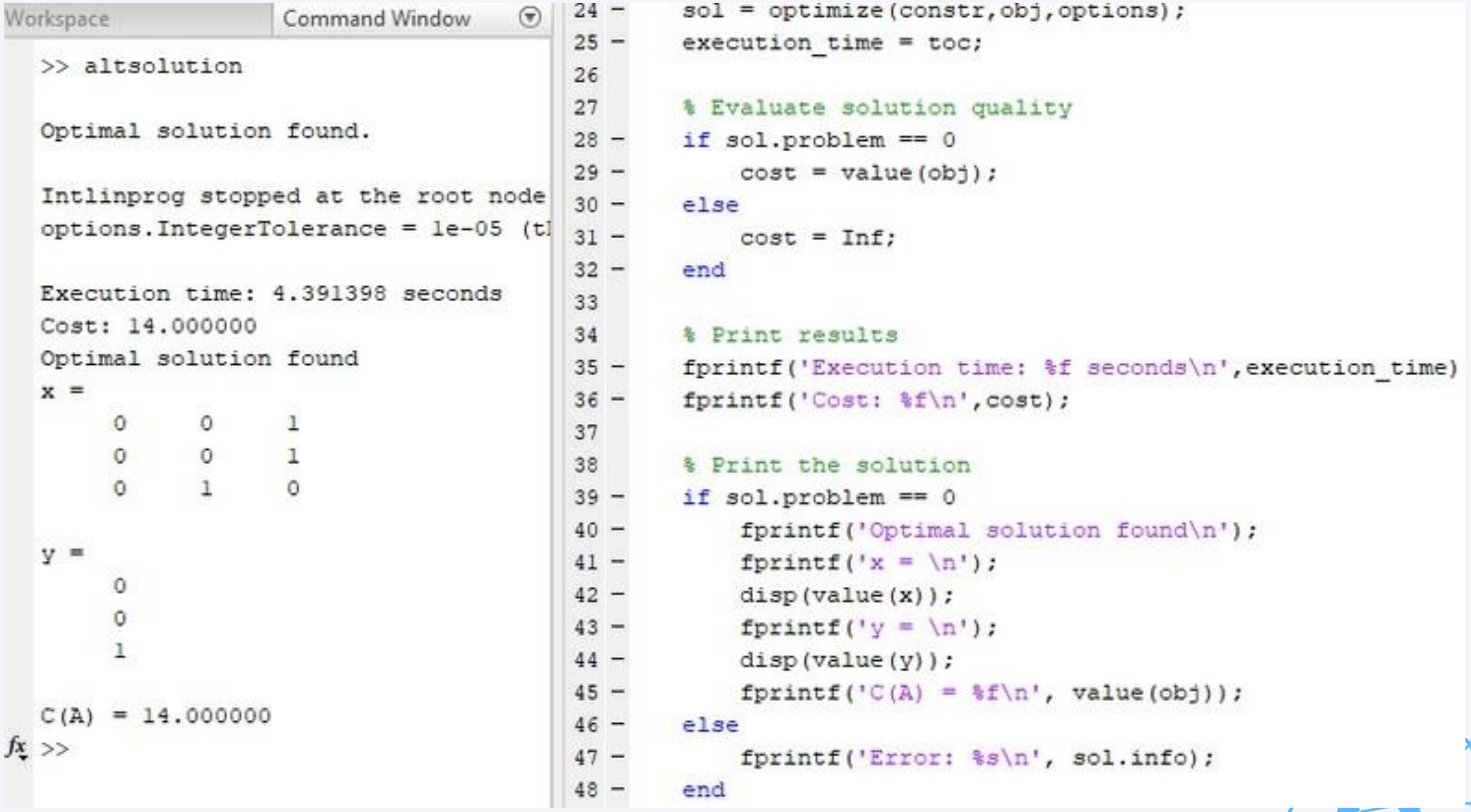


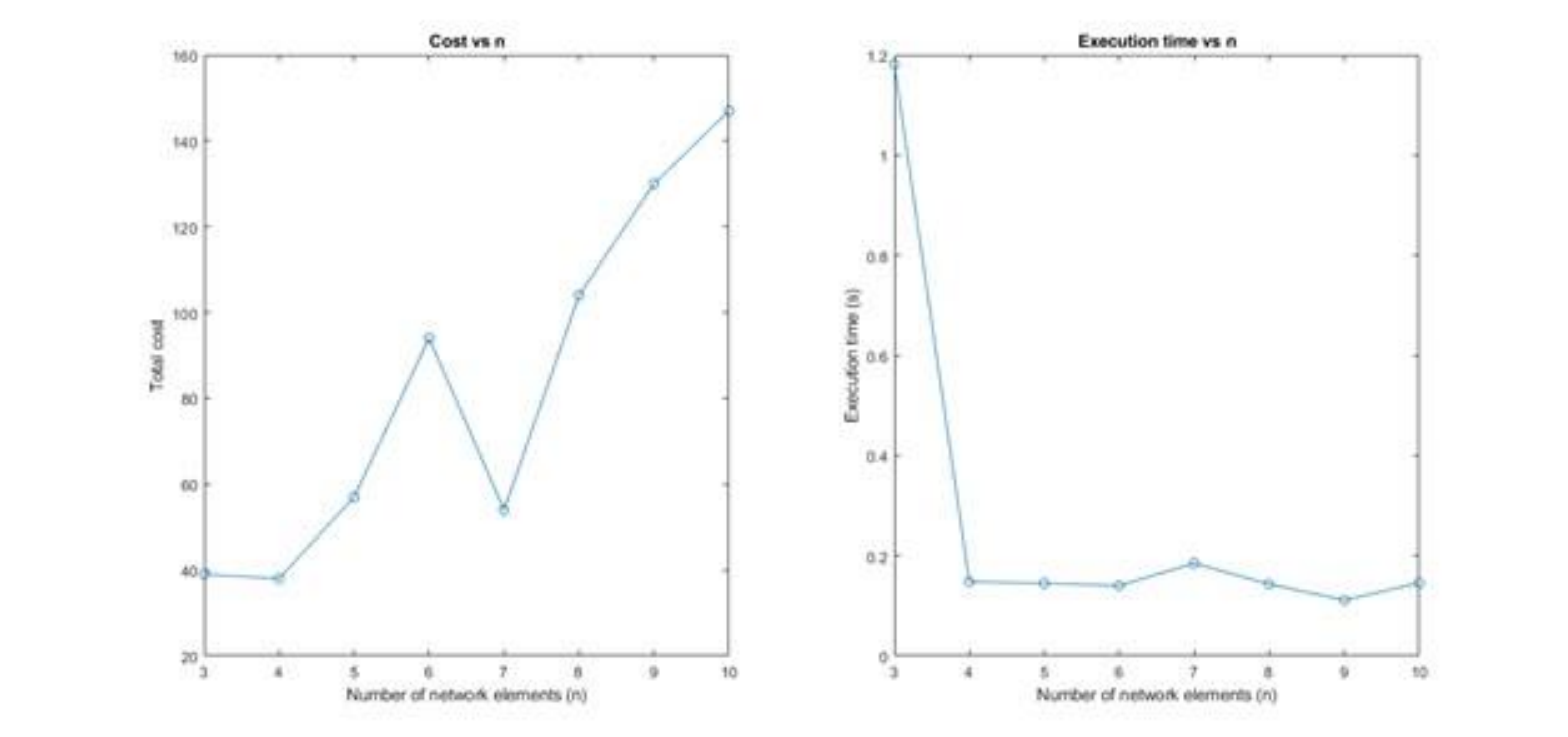
Fig 2: Simulation output

Results & Discussion

- The simulation results show that the proposed approach is effective in optimizing the selection of micro-services and contextual services to minimize the cost while satisfying the latency constraints of the application.

- The number of iterations also increases with the number of network elements, indicating that the algorithm requires more iterations to converge for larger problem sizes.

Results & Discussion



Results & Discussion

- The results obtained further indicate that the execution time decreases with an increasing number of network elements, which suggests that the algorithm scales well for larger networks.

- However, after 4 network elements, the execution time stabilizes, indicating that further increases in network size may not significantly improve the algorithm's performance.

- The cost plot shows a slightly different trend. The cost rises between 4 and 6 network elements, which may be due to the added complexity of the network, before reducing again at 7 elements.

Future Research

Examining the impact of data quality: The results of this study were based on randomly generated data.

However, in real-world scenarios, the quality of the data used to inform optimization algorithms can have a significant impact on the results.

Future research could explore how variations in data quality impact the performance of the ILP algorithm.

Conclusions & Recommendations

- The ILP model is effective in solving the problem of selecting the optimal set of micro-service and contextual service providers for a composite service under latency constraints.

- To further improve the ILP model, it may be worth exploring techniques such as constraint programming or mixed-integer nonlinear programming.

Other Applications

- In context-aware power systems, optimization techniques can be used to determine the most efficient allocation of energy resources to meet user demand while minimizing waste.

- For instance, the use of advanced sensors to monitor energy usage can be combined with optimization algorithms to enable real-time demand-response mechanisms, where energy consumption can be adjusted in response to varying energy prices and user preferences.

References

- Nguyen, D. T., Nguyen, K. K., & Cheriet, M. (2017, October). Optimized IoT service orchestration. In 2017 IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC) (pp. 1-6). IEEE.
- Donassolo, B., Fajjari, I., Legrand, A., & Mertikopoulos, P. (2019, January). Fog based framework for IoT service provisioning. In 2019 16th IEEE annual consumer communications & networking conference (CCNC) (pp. 1-6). IEEE.

Thank You