How to assign client signals to optical carriers?

POLITECNICO
MILANO 1863

NOKIA

Meet the Team

Fatemeh Asadi Tirtashi

Fatemeh.asadi@mail.polimi.it

Kasra Alizadeh

kasra.alizadeh@mail.polimi.it

Table of Contents

Introduction

- **Multi-Carrier Transponder**
  - Converts signals between the optical and electrical domains
- **ODUs (Optical Data Units)**
  - Different ODU types correspond to various data rates, enabling efficient mapping of multiple service types
- **Framers**
  - Modules within the transponder and subject to capacity and ODU-count limits
- **Asymmetric Ports**
  - Ports on a multi-carrier transponder may have different performance capabilities
  - Requires strategic distribution of ODUs to optimize usage across all available ports

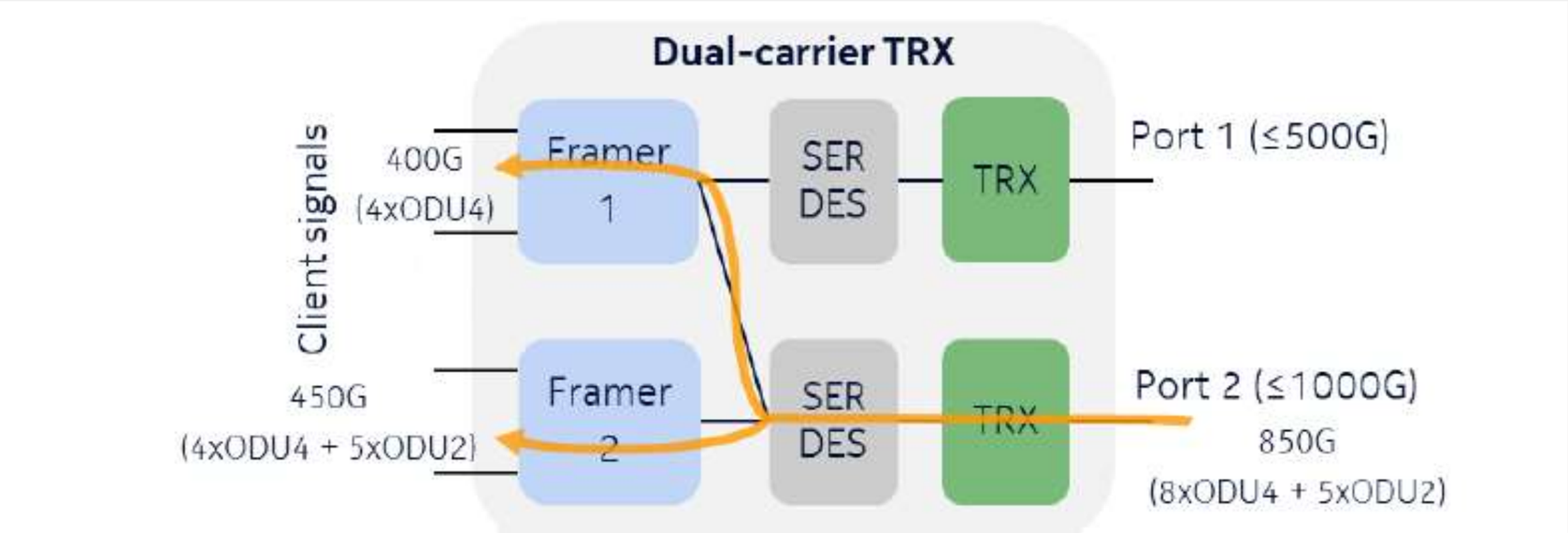


Image source: Nokia, AdoptAProject2024 - details

Problem Statement & Objective

❑ Objective

- Optimize ODU assignment by comparing various optimization methods to ensure:
  - Balanced traffic distribution while respecting capacity limits
  - Efficient utilization of available resources
  - Minimized framer overload and maximized network performance

❑ Problem Constraints

- **Framer Capacity**: Each framer supports up to 500 Gbit/s
- **Framer ODU Limit**: Each framer can handle up to 200 ODUs
- **Optical Signal Capacity**:
  - Port 1 can carry up to 500 Gbit/s
  - Port 2 can carry up to 1000 Gbit/s
- **ODU Types**:
  - ODU0 → 1.25 Gbit/s
  - ODU2 → 10 Gbit/s
  - ODU4 → 100 Gbit/s
  - ODUC4 → 400 Gbit/s

Table 4. Client ODUs

ODU type	Rate, Gbit/s
ODU0	1.25
ODU2	10
ODU4	100
ODUC4	400

Table 5. Framer constraints

Framer	Capacity, Gbit/s	Number of processed ODUs
1	500	200
2	500	200

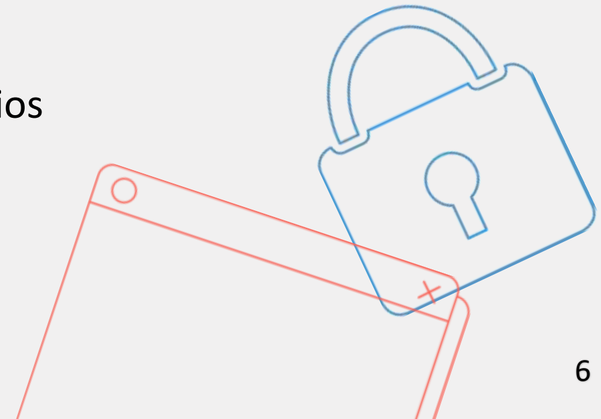
Optimization Methods:

- **Linear Optimization Approach**
  - Integer Linear Programming (ILP)

- **Heuristic & Metaheuristic Approaches**
  - Simulated Annealing (SA)
  - Genetic Algorithm (GA)
  - Particle Swarm Optimization (PSO)
  - Ant Colony Optimization (ACO)
  - Greedy Algorithm

- **Exhaustive Search Approach**
  - Brute Force

- **Why Use Both Approaches?**
  - Linear (ILP): Best for small-scale problems, ensures optimality, but slow for large data
  - Heuristic Methods: Faster, scalable, and find near-optimal solutions for complex real-time scenarios



Integer Linear Programming (ILP)

- **Optimal solution but not scalable Integer Linear Programming (ILP)**

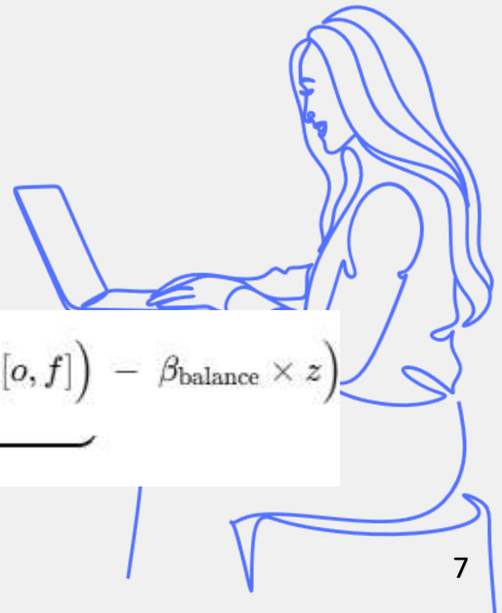
- **ILP Formulation:**

  - ✓ **Decision variables** : Assigns ODUs to framers

  - ✓ **Constraints** :
    - Framer Capacity: Each framer supports up to 500G
    - ODU Limit: A framer can process at most 200 ODUs
    - Traffic Demand: All ODUs must be assigned correctly

  - ✓ **Objective Function**
    - Maximize assigned ODUs without exceeding framer capacity
    - Uses λ (lambda) to prioritize traffic allocation and β (beta) to penalize load imbalance

$$\max\Big(\underbrace{\sum_{o\in ODUs}\Big(x[o,\text{Framer1}]\cdot \text{ODU\_rates}[o]\Big)+\sum_{o\in ODUs}\Big(x[o,\text{Framer2}]\cdot \text{ODU\_rates}[o]\Big)}_{\text{total assigned traffic}}-\lambda_{\text{penalty}}\times\underbrace{\sum_{o\in ODUs}\Big(\text{traffic}[o]-\sum_{f\in Framers}x[o,f]\Big)}_{\text{unassigned traffic}}-\beta_{\text{balance}}\times z\Big)$$



Brute Force

- **Guarantees an optimal solution by exhaustively exploring all possibilities**

- **Brute Force Formulation:**
  - Generate all possible ODU-to-framer assignments
  - For each assignment, verify framer capacity (≤500 Gbit/s) and ODU limit (≤200 ODUs)
  - Compute total allocated traffic for valid assignments
  - Select the assignment that maximizes traffic allocation

Genetic Algorithm (GA)

❑ **GA formulation**:

- Gene: A single gene indicates which framer a specific ODU is assigned to. Each gene holds the “framer ID,” telling us which framer is processing that ODU
- Chromosome: A complete solution—a list of genes detailing assignments for all ODUs (If we have 10 ODUs and 2 framers, each chromosome is a 10-gene sequence. A gene value of “1” might mean “ODU goes to Framer 1,” while “2” means “ODU goes to Framer 2)
- Fitness Function: Maximizes assigned traffic without exceeding 500 Gbit/s per framer
- Genetic Operators:
- Selection: Chooses high-fitness chromosomes to pass on their traits
- Crossover: Swaps gene segments between two chromosomes to create new solutions
- Mutation: Randomly alters genes to maintain population diversity

Simulated Annealing (SA)

❑ **A metaheuristic inspired by metal cooling that gradually refines solutions**

❑ SA Formulation :

- Start with a random ODU assignment
- Evaluate the total assigned traffic
- Make small changes (e.g., swap ODUs between framers)
- Accept or reject changes based on a probability [P(accept)=exp(Δ/T)]controlled by a temperature function
- After each iteration, reduce the temperature:
  - Begins high (encouraging exploration) and gradually decreases (focusing on refinement)
  - Update rule: T=T×0.95T (higher T favors exploration, lower T favors exploitation)
- Stopping Criterion: Continue until TTT is very low, or after a fixed number of iterations with no improvement

Particle Swarm Optimization (PSO)

- **Inspired by swarm intelligence, PSO models ODUs as particles searching for an optimal assignment**

- **PSO Formulation**:

- Each particle represents an ODU assignment
- Particles update based on the best-known solutions
- Velocity adjustments guide particles toward better solutions
- Converges to an optimal assignment over iterations

Ant Colony Optimization (ACO)

❑ **A metaheuristic inspired by ants' pheromone trails, where stronger pheromones reinforce good ODU assignments**

❑ ACO Formulation:

- Ants explore random ODU assignments
- Pheromones reinforce better solutions
- Ants follow stronger pheromone trails
- Iterate until an optimal assignment is found

Greedy Algorithm

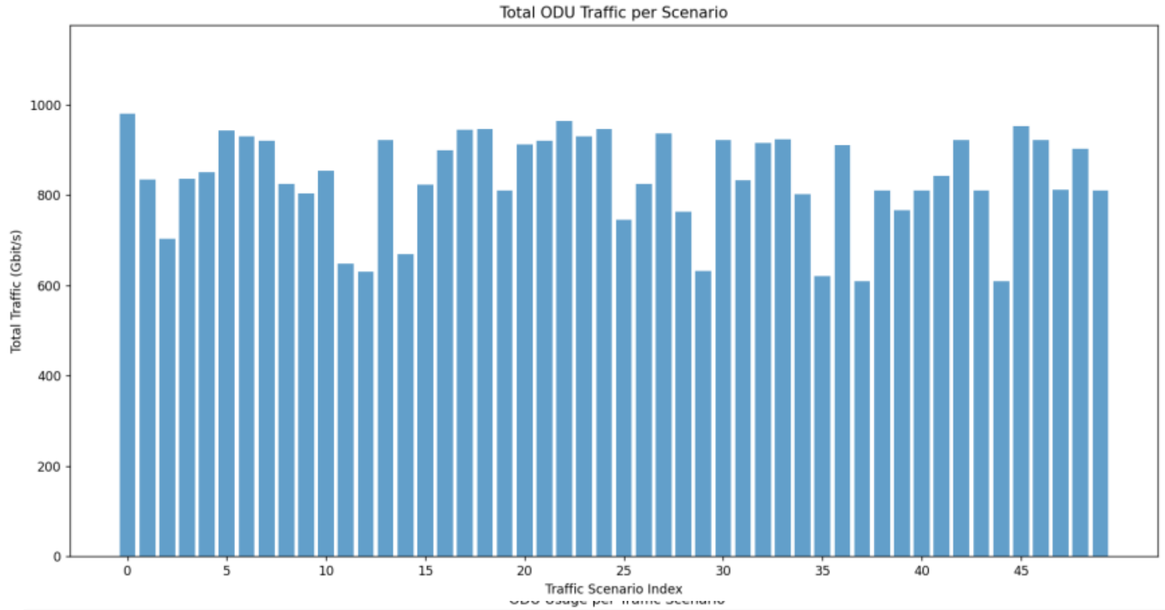
❑ **A heuristic method that assigns ODUs to framers one by one based on available capacity**

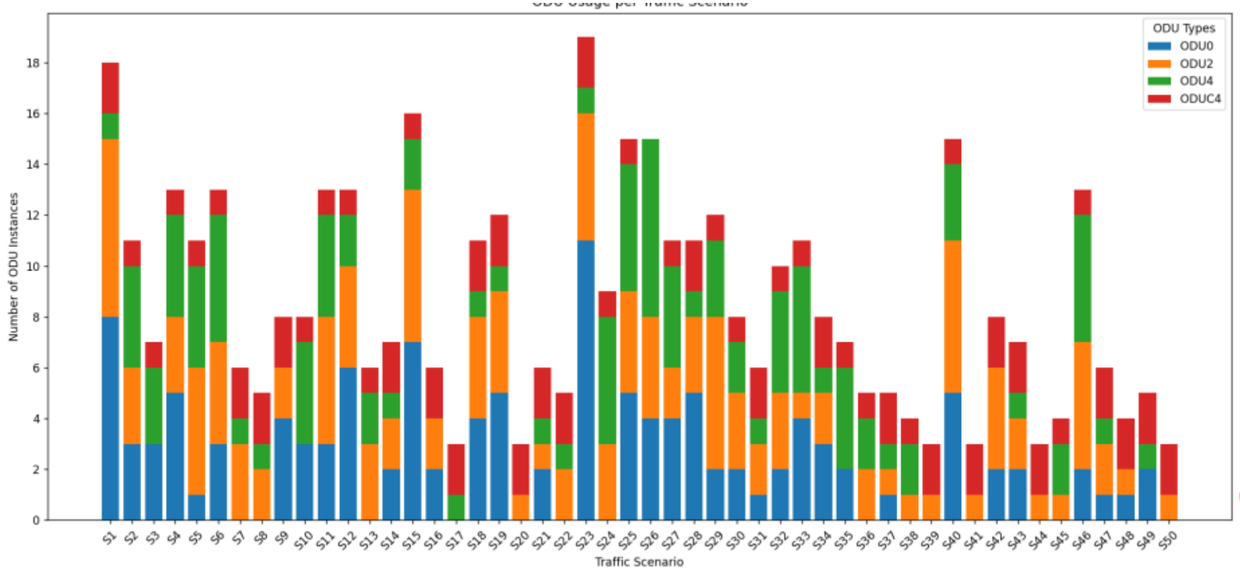
❑ **Greedy Formulation:**

- Sort ODUs by size (largest first)
- Assign ODUs to the framer with the least current load
- Check capacity constraints (≤500G per framer)
- Stop when all ODUs are processed

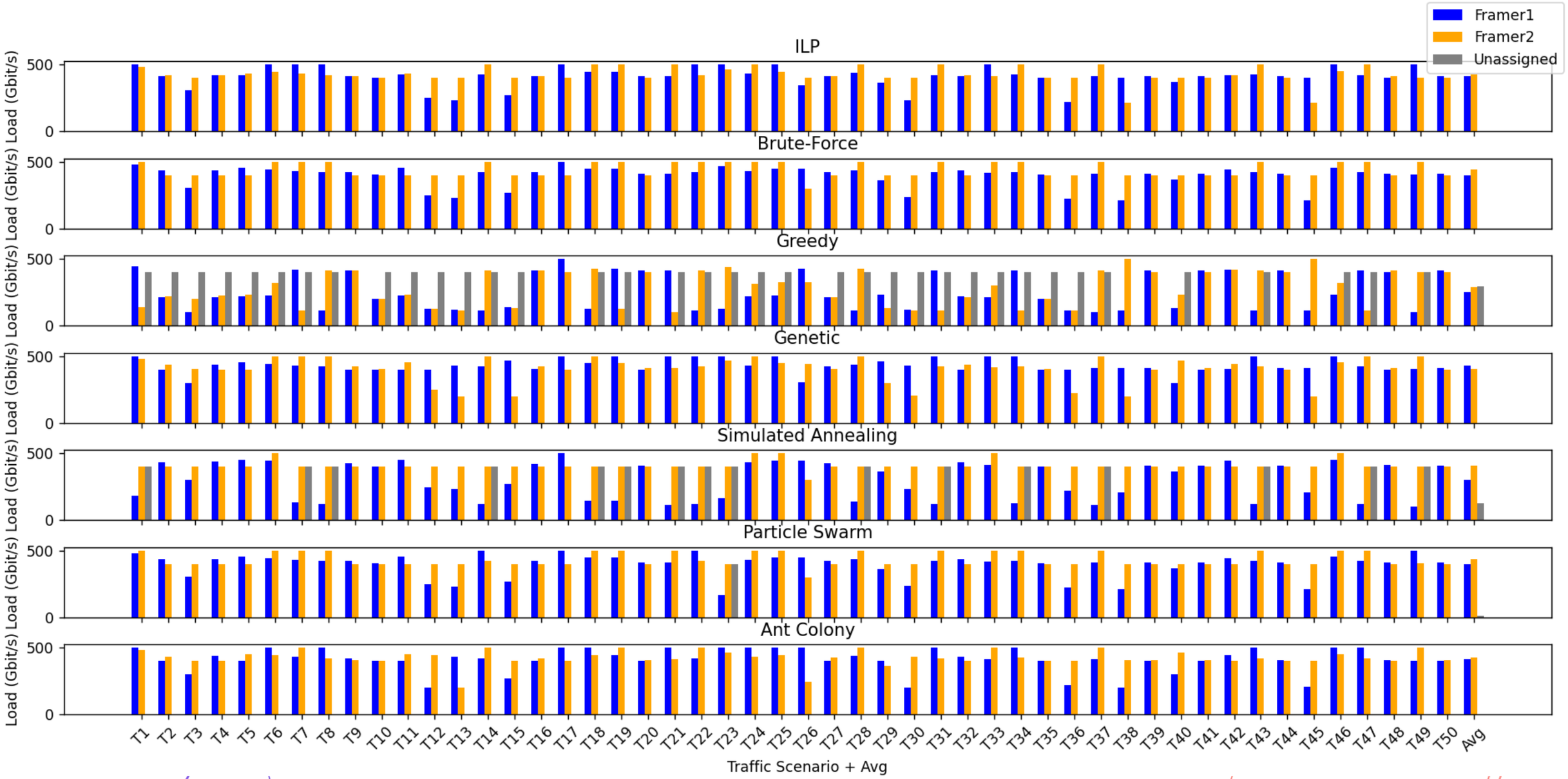
Simulation set up & numerical results

❑ **Traffic Scenarios:**
- Different traffic loads and patterns
- Traffic is considered from **Port 2** of the transponder (up to 1000 Gbit/s)

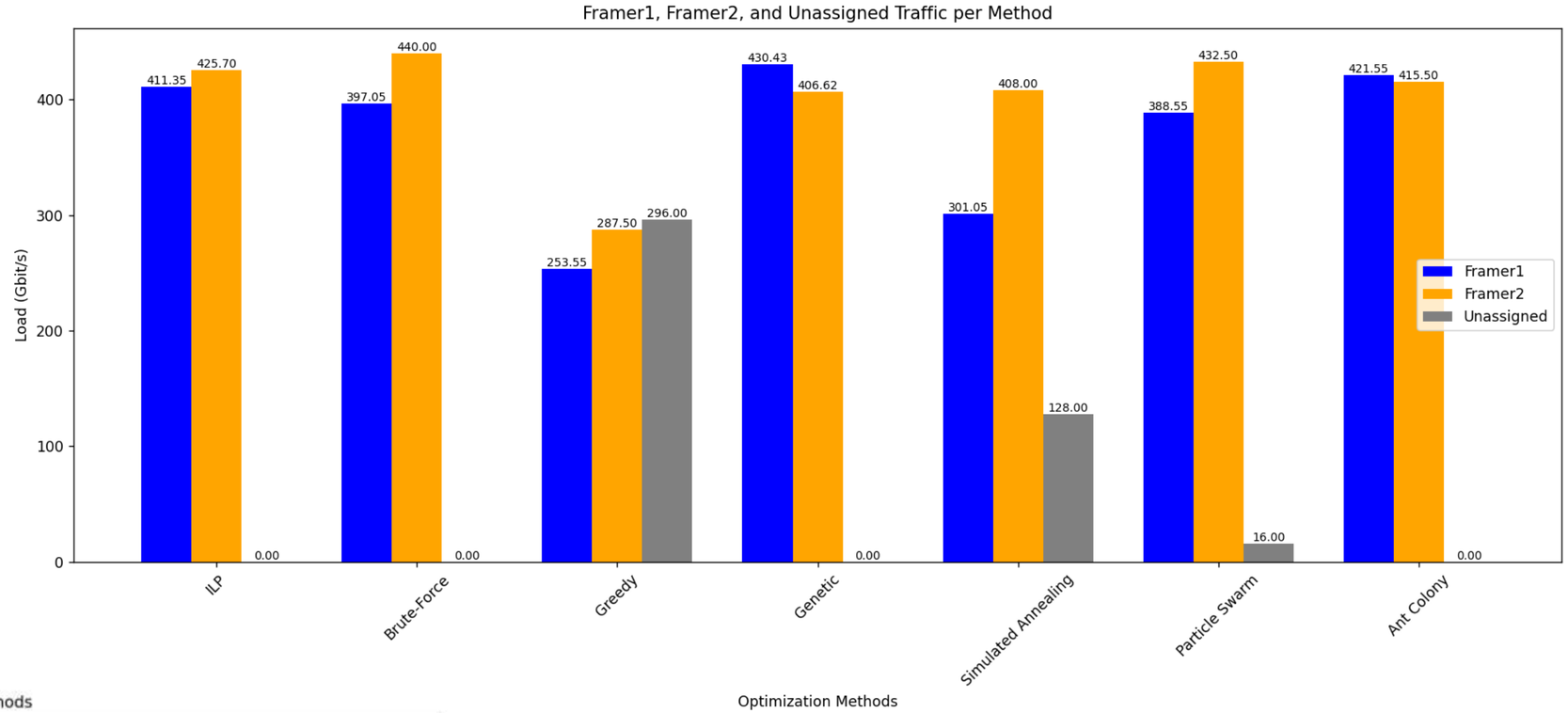


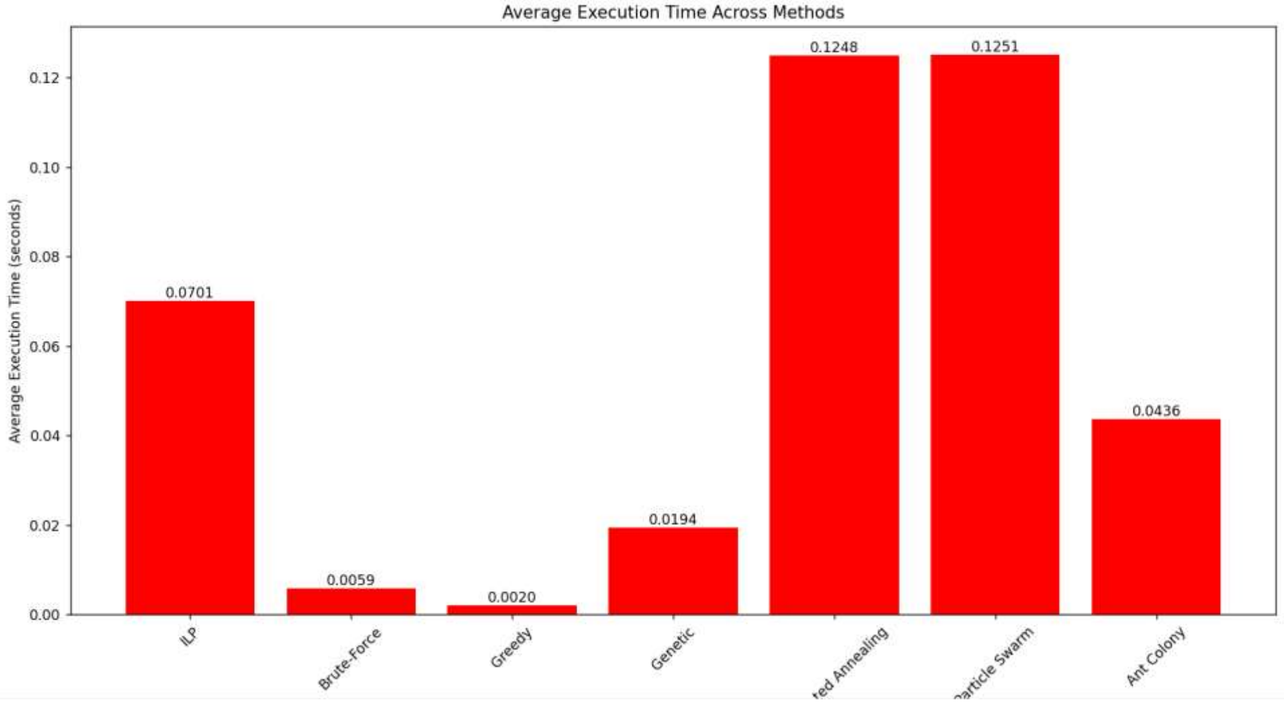


Numerical Result



Comparison


Framer1, Framer2, and Unassigned Traffic per Method
Load (Gbit/s)
0
100
200
300
400
411.35
425.70
0.00
397.05
440.00
0.00
253.55
287.50
296.00
430.43
406.62
0.00
301.05
408.00
128.00
388.55
432.50
16.00
421.55
415.50
0.00
ILP
Brute-Force
Greedy
Genetic
Simulated Annealing
Particle Swarm
Ant Colony
Framer1
Framer2
Unassigned
Optimization Methods


Average Execution Time Across Methods
Average Execution Time (seconds)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.0701
0.0059
0.0020
0.0194
0.1248
0.1251
0.0436
ILP
Brute-Force
Greedy
Genetic
Particle Swarm
Ant Colony

Comparison

Approach	Execution Time (Avg)	Unassigned Traffic	Optimality	Scalability
ILP (Integer Linear Programming)	Low (0.0701 s)	Very Low	Optimal (small scale)	Limited for larger problems
Brute Force	Low (0.0059 s)	Very Low	Optimal (small scale)	Not scalable (exponential growth)
Greedy	Very Low (0.002 s)	High	Suboptimal	Moderate (fast but naive)
Genetic Algorithm (GA)	Moderate (0.0194 s)	Moderate	Near-optimal	Good (parameter tuning required)
Simulated Annealing (SA)	High (0.1251 s)	Moderate	Near-optimal	Good (sensitive to schedule)
Particle Swarm Optimization (PSO)	High (0.1248 s)	Moderate	Near-optimal	Good (sensitive to parameters)
Ant Colony Optimization (ACO)	Moderate (0.0436 s)	Low–Moderate	Near-optimal	Good (depends on parameter tuning)

Conclusion

Method	Speed	Unassigned Traffic	Optimality	Scalability
ILP	★★☆☆☆	★★★★★	★★★★★	★★☆☆☆
Brute Force	★★★★☆	★★★★★	★★★★★	★☆☆☆☆
Greedy	★★★★★	★☆☆☆☆	★★☆☆☆	★★★★☆
GA	★★★★☆	★★★★☆	★★★★☆	★★★★☆
SA	★★☆☆☆	★★★☆☆	★★★★☆	★★★★☆
PSO	★★☆☆☆	★★★☆☆	★★★★☆	★★★★☆
ACO	★★★☆☆	★★★★☆	★★★★☆	★★★★☆

❑ **No single algorithm is perfect. The best choice depends on the trade-off between speed, accuracy, and scalability!**

❑ Future Work:

- Hybrid Approaches → Combining ILP with heuristics for improved efficiency
- Real-Time Adaptation → Dynamic optimization for changing network demands
- Scalability Improvements → Enhancing ACO and PSO for better large-scale performance

THANK YOU!

Special thanks to :

Prof. Massimo Tornatore

Aryanaz Attarpour

Prof. Mëmëdhe Ibrahimi

and Nokia Company

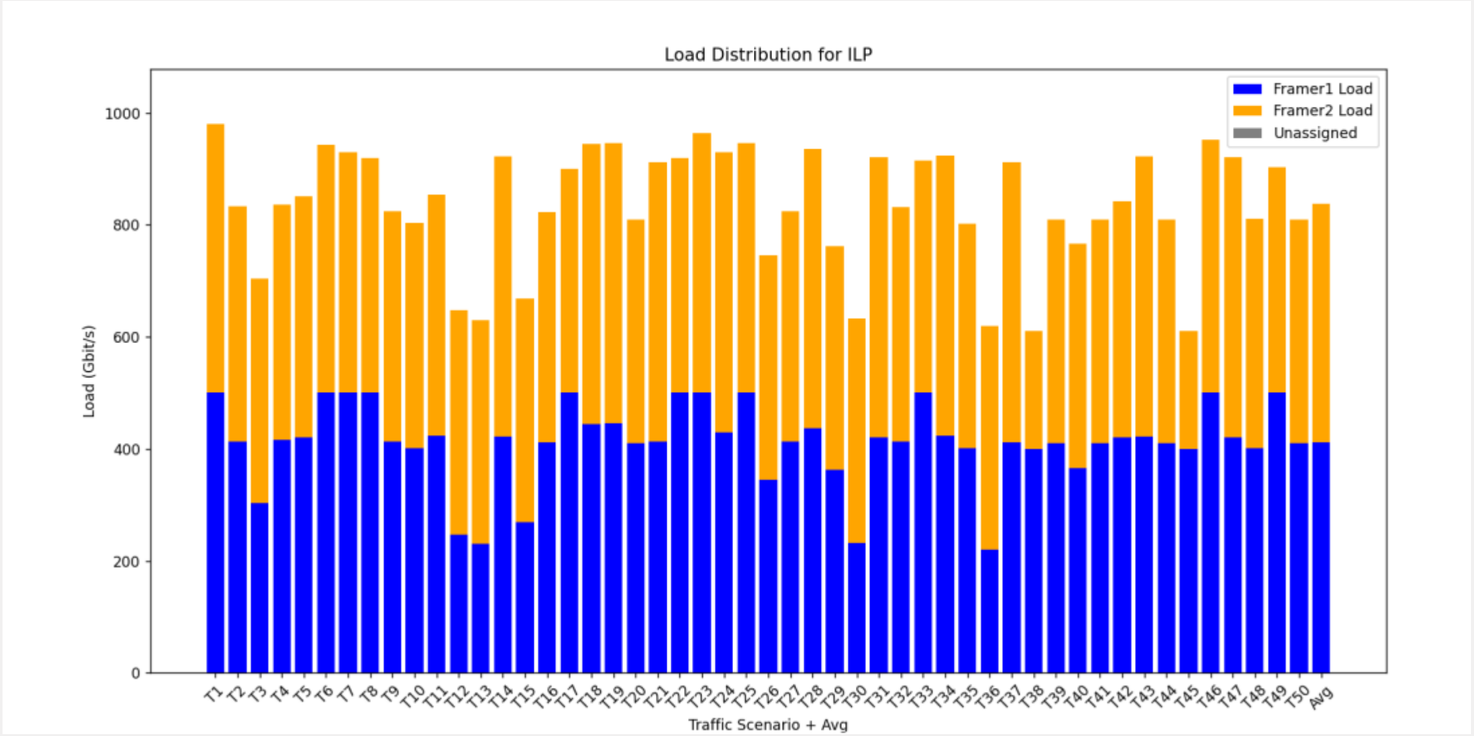
Back up slide

ILP

Results Overview (Using Gorubi Solver)

- ILP efficiently assigns ODUs, balancing load between Framer 1 & 2
- Graph shows load distribution across 50 scenarios + average
- ILP adapts dynamically to varying traffic patterns

✓ Pros: Ensures optimal ODU allocation, prevents overload, and adapts to demand
✗ Cons: Computationally expensive for large networks; strict constraints may leave some ODUs unassigned

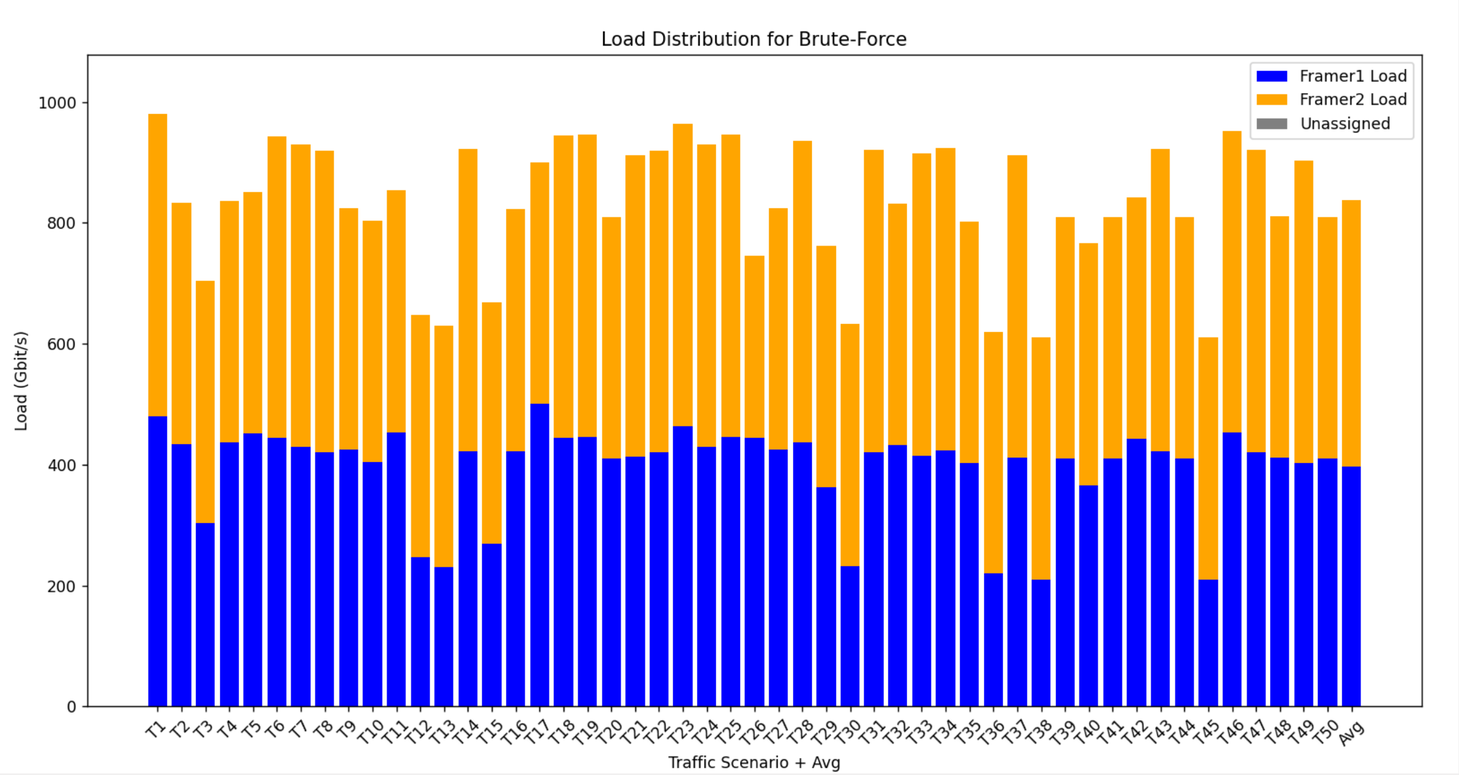


Brute Force

Results Overview
- Brute-force explores all assignments but struggles with large-scale scenarios
 - Optimal for small cases but fails in complex scenarios
- Computation time grows exponentially, making it impractical for real-time use

✓ Pros:
- Always finds the best solution when feasible
- Useful for benchmarking optimization methods

✕ Cons:
- Extremely slow for large-scale networks
- Fails under strict constraints, leaving traffic unassigned

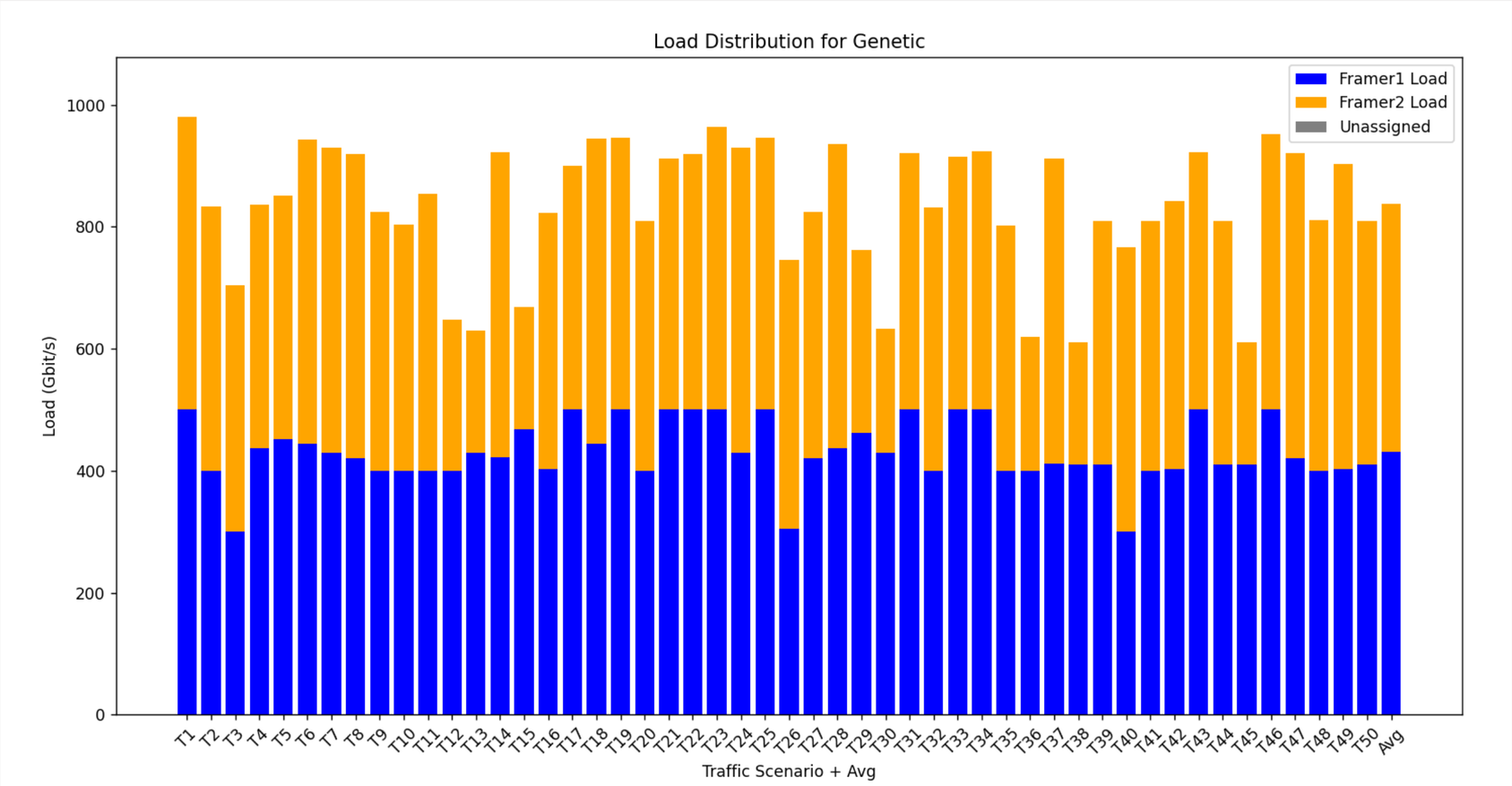


Genetic Algorithm (GA)

Genetic Algorithm (GA) - Results
- GA achieves better traffic allocation than Greedy
- Framer utilization is more balanced

✓ Pros:
- Better optimization than greedy
- Adapts dynamically to traffic variations

✗ Cons:
- Not guaranteed to be optimal
- Slower than simpler methods

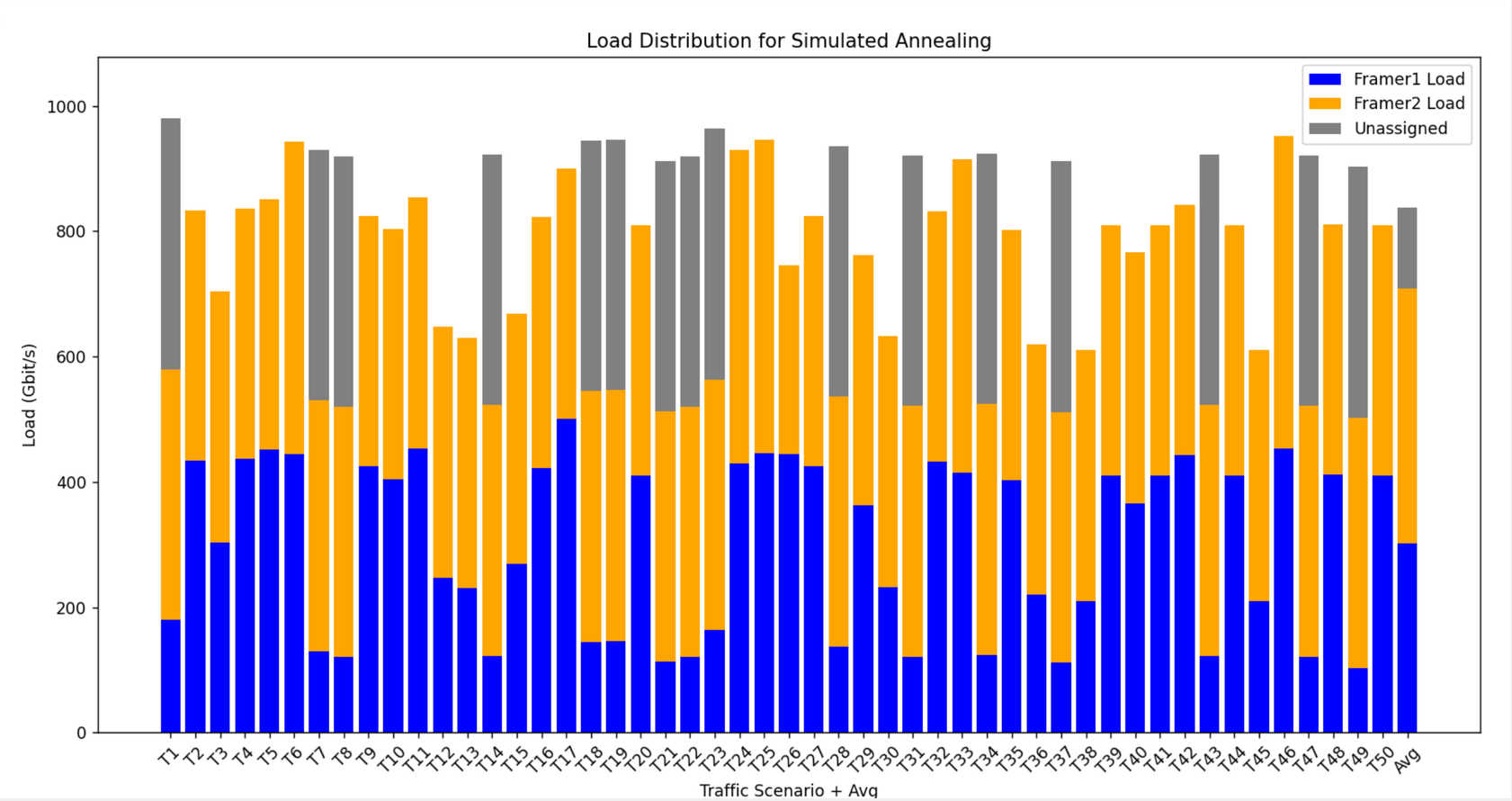


Simulated Annealing (SA)

Simulated Annealing (SA) - Results
- SA improves allocation compared to GA, but unassigned traffic remains an issue.
- Some scenarios show better framer balance, but overall inefficiency persists.

✓ Pros:
- Better optimization than greedy
- Finds good solutions in complex scenario

✕ Cons:
- Slower than greedy.
- May not always outperform GA



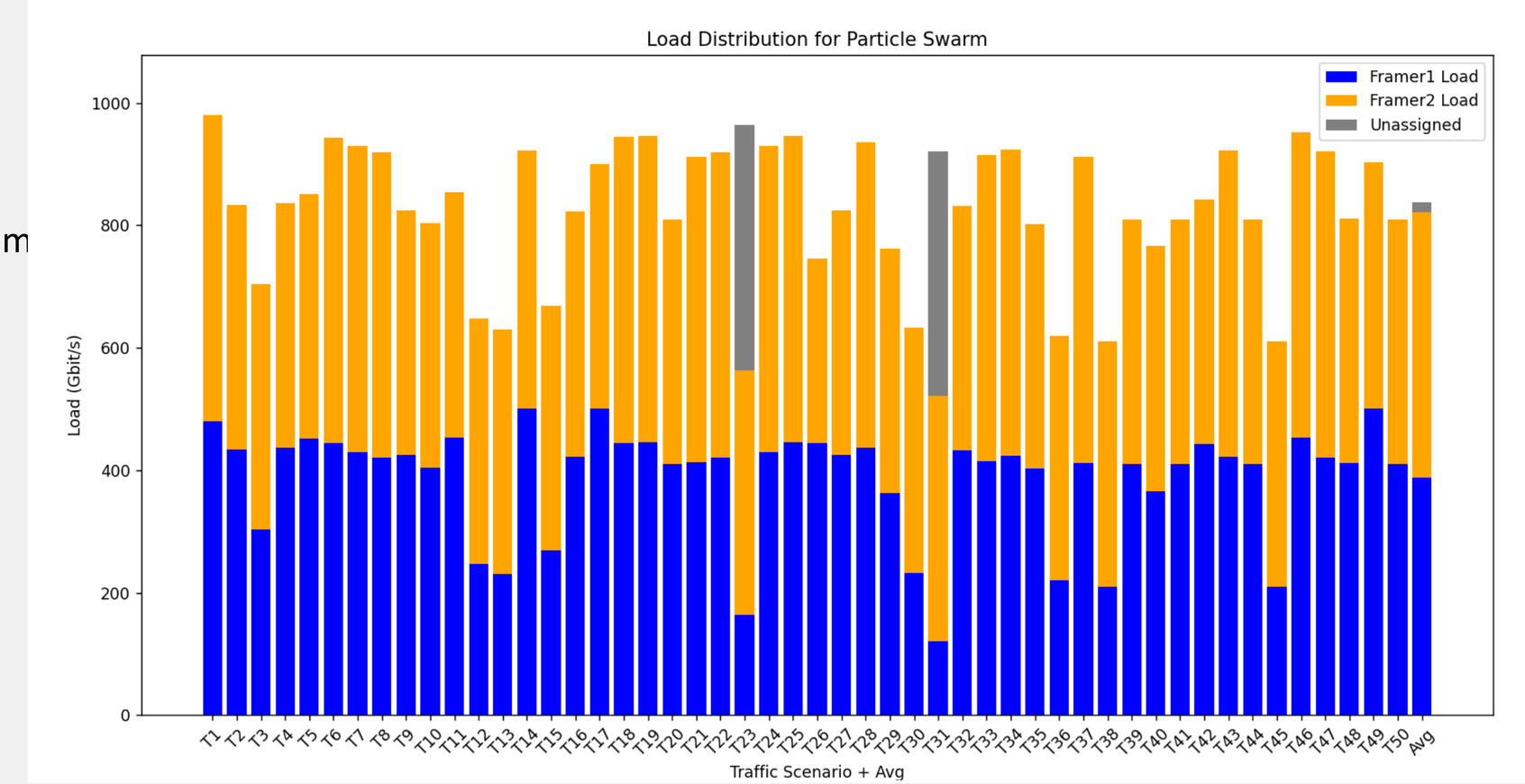
Particle Swarm Optimization (PSO)

PSO - Results

- PSO shows improvements over Greedy, with better traffic distribution
- Still, some unassigned traffic remains in specific scenarios

✓ Pros:
- Efficient and adaptive
- Better than basic heuristics like greedy

✕ Cons:
- Requires tuning for best performance.
- Can struggle with highly constrained problem



Ant Colony Optimization (ACO)

ACO - Results

- ACO provides a balanced traffic allocation

- Performs better than Greedy and SA, but not always as good as ILP

✓ Pros:

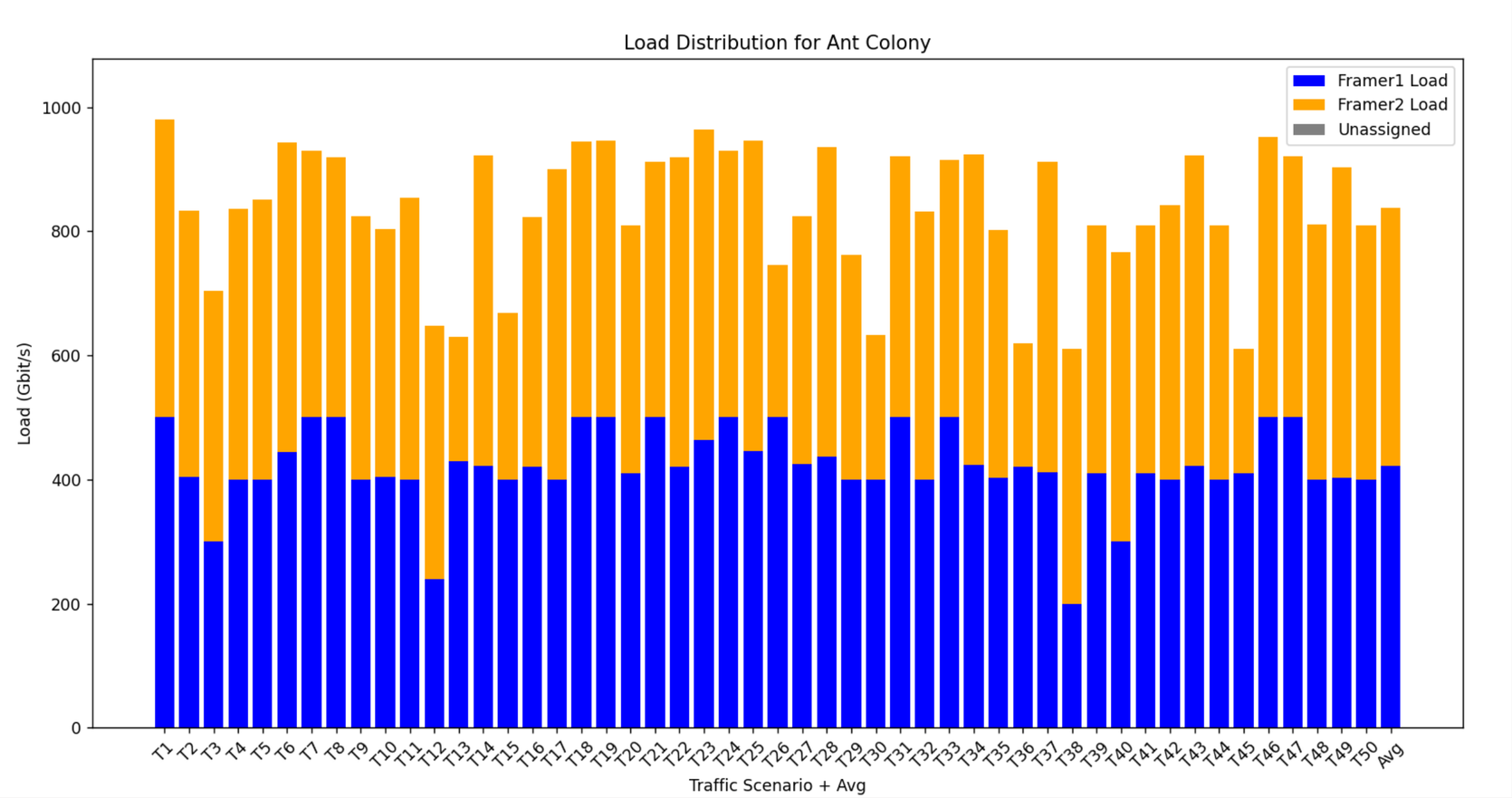
- Self-improving algorithm

- Performs well in large-scale scenarios

✕ Cons:

- Computationally expensive

- Slow convergence



Greedy Algorithm

Greedy Algorithm - Results
- Greedy algorithm assigns traffic quickly but leads to high unassigned traffic
- Traffic distribution between Framer 1 and Framer 2 is unbalanced
- Many ODUs remain unassigned due to the algorithm’s limited lookahead

✓ Pros:
- Fast and simple, good for quick allocation
- Works well in lightly loaded networks

✕ Cons:
- Not optimal, leading to imbalanced resource usage
- High unassigned traffic, reducing efficiency