Routing Optimization for MCQF in Time-Sensitive Networking

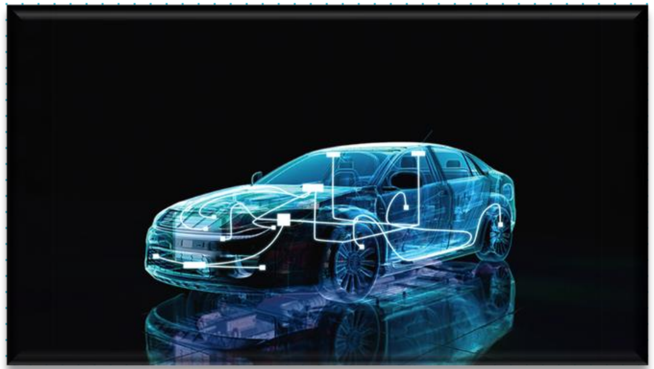
-Parth Anand (parth.anand@tum.de)

-MSc. Student, Technical University of Munich

Why TSN?

Applications:


Industrial Automation


In-Vehicle Networking


Avionics


Healthcare

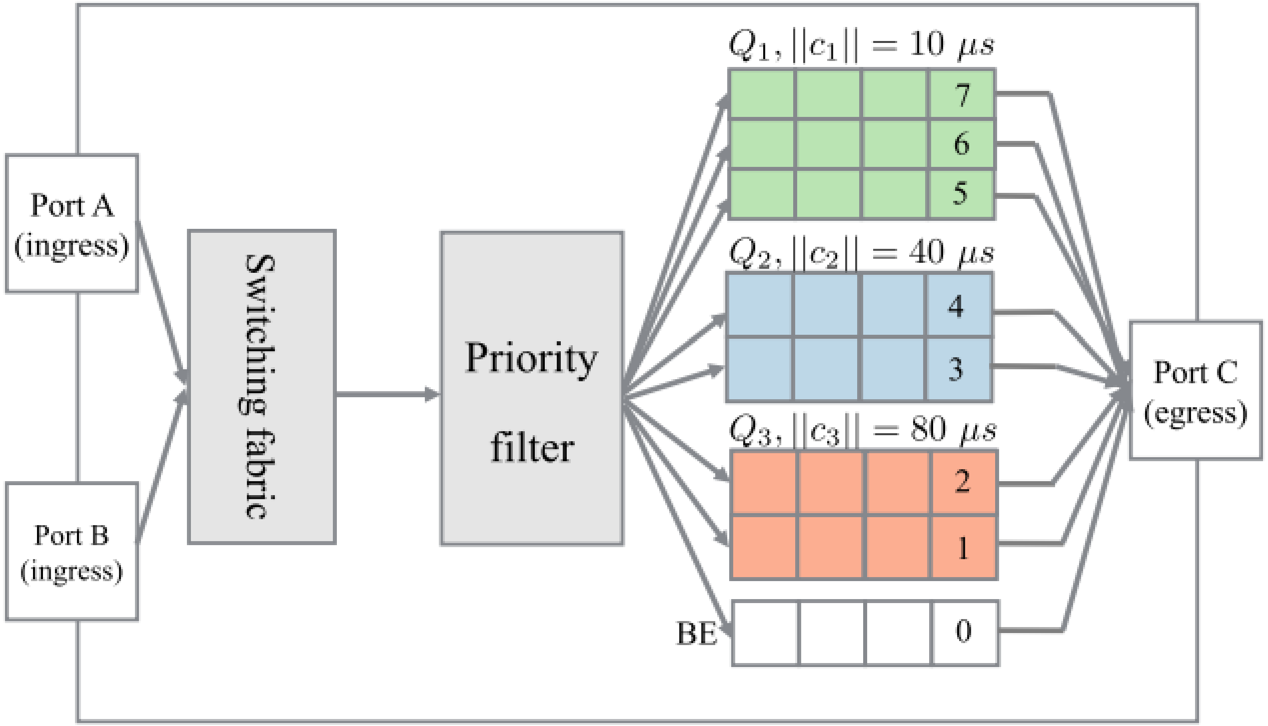

Real time Audio/Video transport

What is TSN?

- Deterministic Networking over Ethernet.
- Set of IEEE standards
- IEEE 802.1AS: Time Synchronization
- IEEE 802.1Q: Priority Handling :
  - IEEE 802.1Qbv: Time Aware Shaper
  - IEEE 802.1Qch: MCQF
  - IEEE 802.1Qav: Credit based shapers
  - IEEE 802.1Qbu: Frame preemption

Multiple Cyclic Queuing and Forwarding (MCQF):

- 802.1Qch
- Traffic scheduling mechanism
- real-time and bounded worst-case latencies



Scheduling and Routing:

- High time complexity of the scheduling problem leads to separate the routing problem.
- As an input, scheduling approaches require specifications of flows and routes.
- Off the shelf algo: Shortest path routing
- Existing research has not explored impact of different routing algorithms on the schedulability of the set of flows

Received Codebase:

- Scheduling algorithm already implemented with GA and SA.
- Use of Shortest Path Algorithm for routing.
- Very less schedulability percentage.
- Reason is bandwidth violation.
- Need of load balancing routing approach.
- Literature survey.

Literature Survey:

- Routing Algorithms for IEEE802.1Qbv Networks
- Routing Heuristics for Load-balanced Transmission in TSN-Based Networks.

LBDRR Algorithm:

Algorithm 1: LB-DRR routing scheme.

```
Data: Network topology G; Set of flows F; Constant K
Result: List of best routes for each flow in F
R ← empty list[];
edges ← Set of all edges in G;
load ← zeros[|edges|];
foreach f_i ∈ F do
    R_i ← [];
    Compute r_i = Best(f_i) (see Equation 4);
    foreach edge ∈ r_i do
        load[edge] = load[edge] + C_i;
    end
    R_i.append(r_i);
    if rep_i > 0 then
        used_edges ← {edge ∈ r_i};
        routes = valid_routes(G, src_i, dst_i);
        for j = 1 to rep_i do
            r_{i,j} = arg min_{r∈routes} (|used_edges ∩ {edge ∈ r}|);
            foreach egde ∈ r_{i,j} do
                load[edge] = load[edge] + C_i;
            end
            used_edges = used_edges ∪ {edge ∈ r_{i,j}};
            R_i.append(r_{i,j});
        end
    end
    R.append(R_i);
end
return R
```

Steps followed:

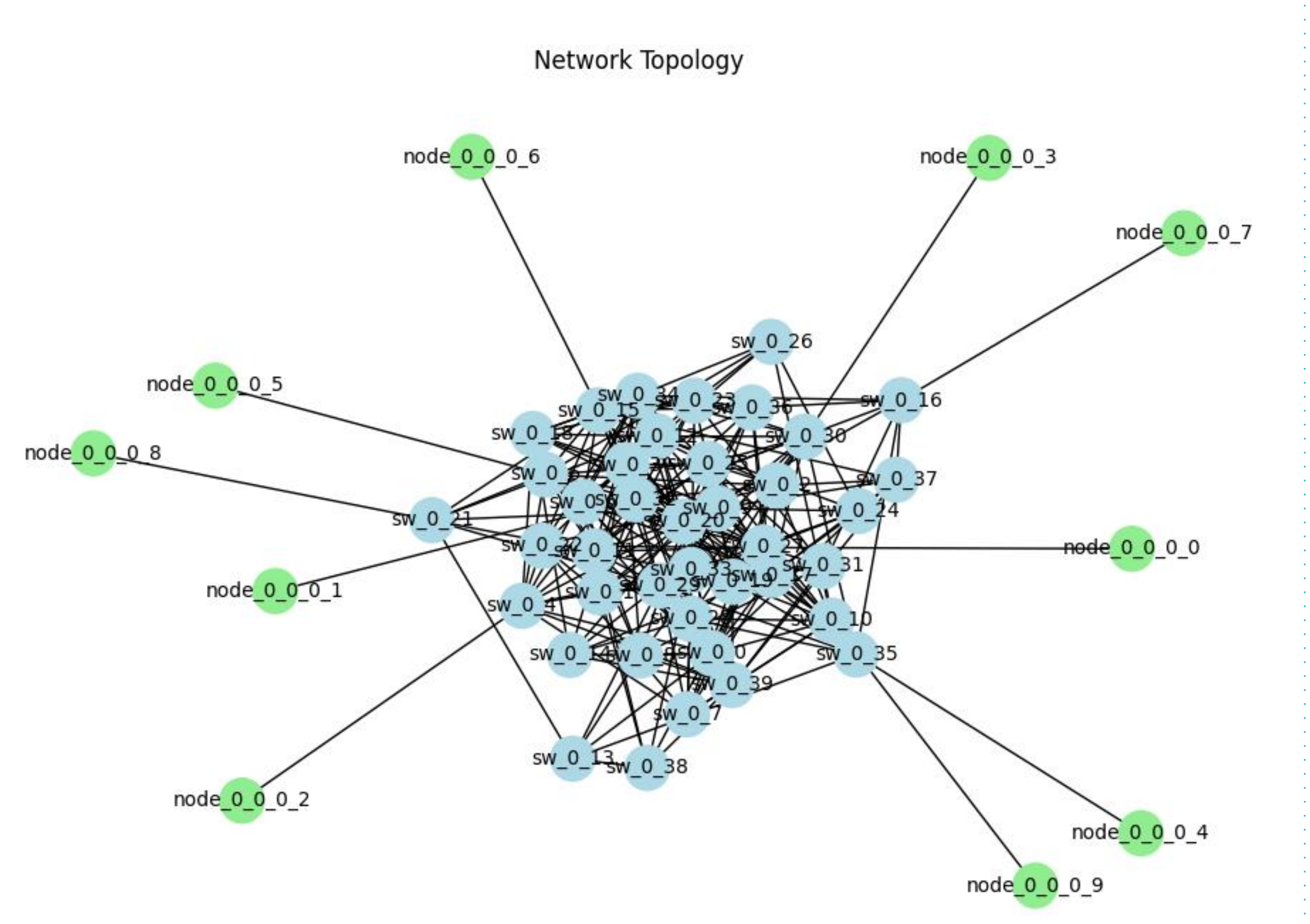
- Modified and implemented algorithm
- Deadline-Aware Path Filtering
- Use of different cost functions

Steps followed:

1. Flow Sorting
2. Topology Creations
3. Readings and plots

- **Topology:**

-Industrial Topology
-n = 50
-PLCs= 10
-Switches= 40
- Link Bandwidth= 1Gbps



Formulas and Constraints:

- 1. $\boldsymbol{r_i} = ((src_i, v_1), (v_1, v_2), \ldots, (v_{n-1}, dst_i))$
- 2. $\boldsymbol{load}(\boldsymbol{e}) = \sum(size(f_j)/period, \forall\ f_j\ traversing\ e$
- 3. $\boldsymbol{Max\_Load}(\boldsymbol{r_i})$ = max_$load(e), \forall\ e\ in\ L_{intermediate}$
- $\boldsymbol{L_{intermediate}} = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-2}, v_{n-1})\}$
- 4. $\boldsymbol{Total\_Load}(\boldsymbol{r_i}) = \sum load(e), \forall\ e\ in\ r_i$
- 5. $\boldsymbol{Average\_Load}(\boldsymbol{r_i}) = \frac{Total\_Load(r_i)}{|r_i|}$
- 6. **Cost**$(\boldsymbol{r_i}) = \propto * Max\_Load(r_i) + \beta * Total\_Load(r_i) + \gamma * Average\_Load(r_i)$

Formulas and Constraints:

- **Deadline Constraint**: $delay(f_i) \leq D_i$
- **Bandwidth Constraint**: $load_{PGk}(e) \leq B_{PGk}$
- **Schedulability**:
- $\boldsymbol{Schedulable(f_i)} = \begin{cases} 1, if\, delay(f_i) \leq D_i \text{ and } load_{PGk}(e) \leq B_{PGk} \\ 0, otherwise \end{cases}$

Common parameters:

1. Link Bandwidth=1000e6 (1Gbps)

2. n=50(number of nodes)

3. # switches =40 (80%)

4. # PLCs= 10 (20%)

5. n_neighbours (for load balancing algo) = 300

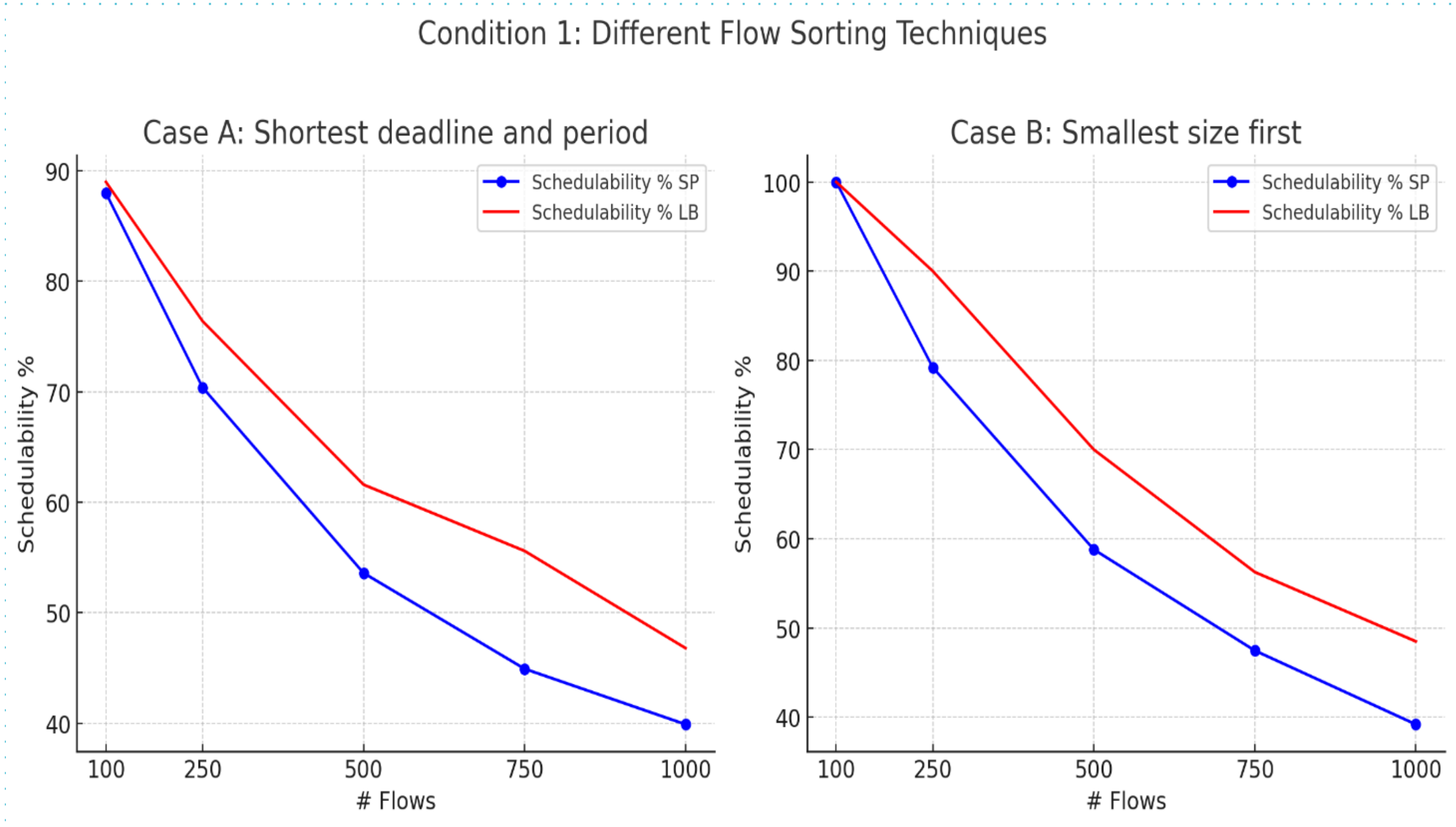
6. Cost function: Max load (with skipping 1st and last link)

Results: Condition 1

Other constant Data Attributes (Common for both cases A and B):

1. Deadline and Period= [1000,2000,5000,10000]

2. Flow Size= random(100-1500) bytes

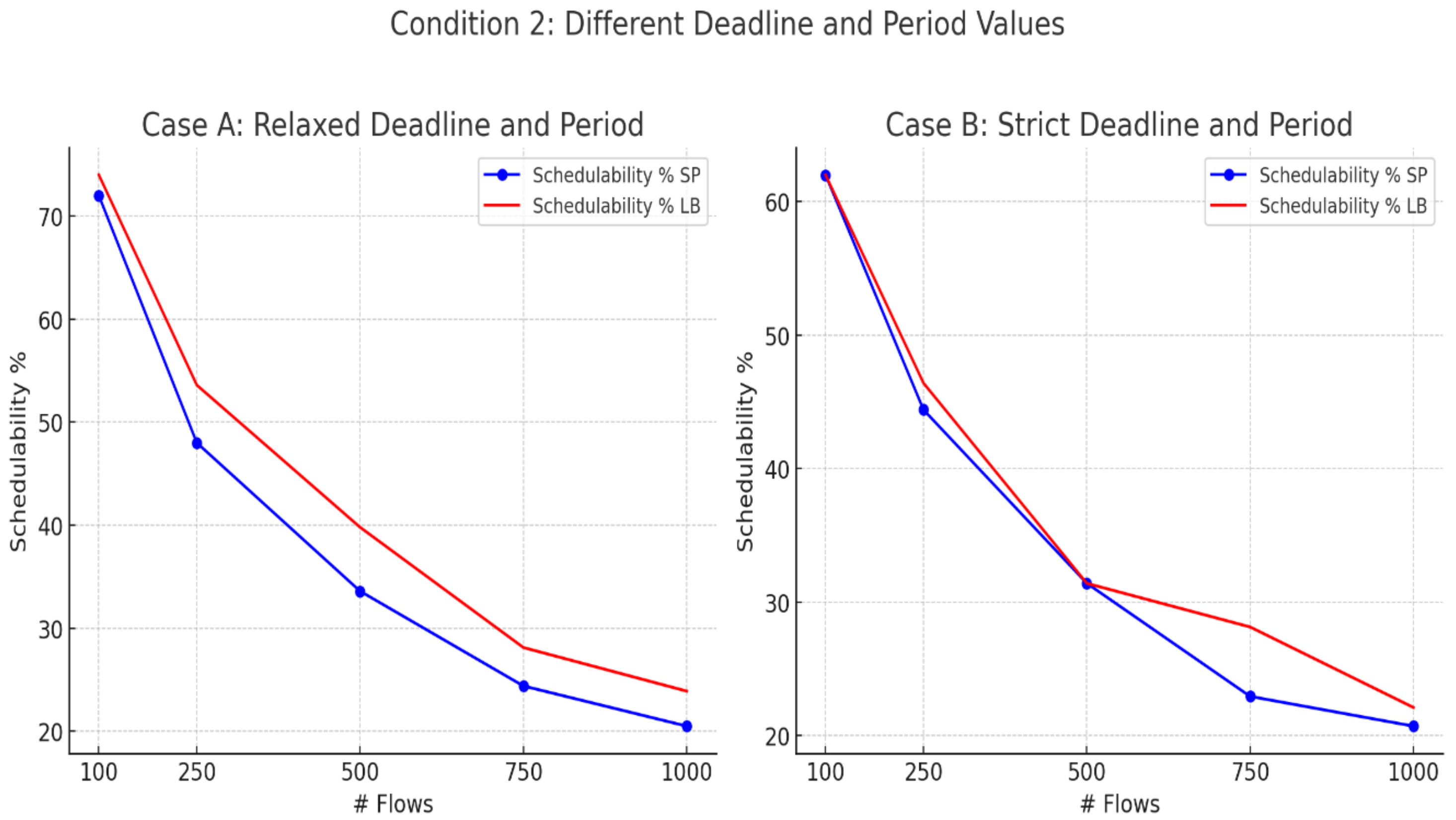


Results: Condition 2

Other constant values:

1. Sorting of flows: Shortest deadline and period first (strict)

2. Flow Size= random(1000-1500) bytes (strict)

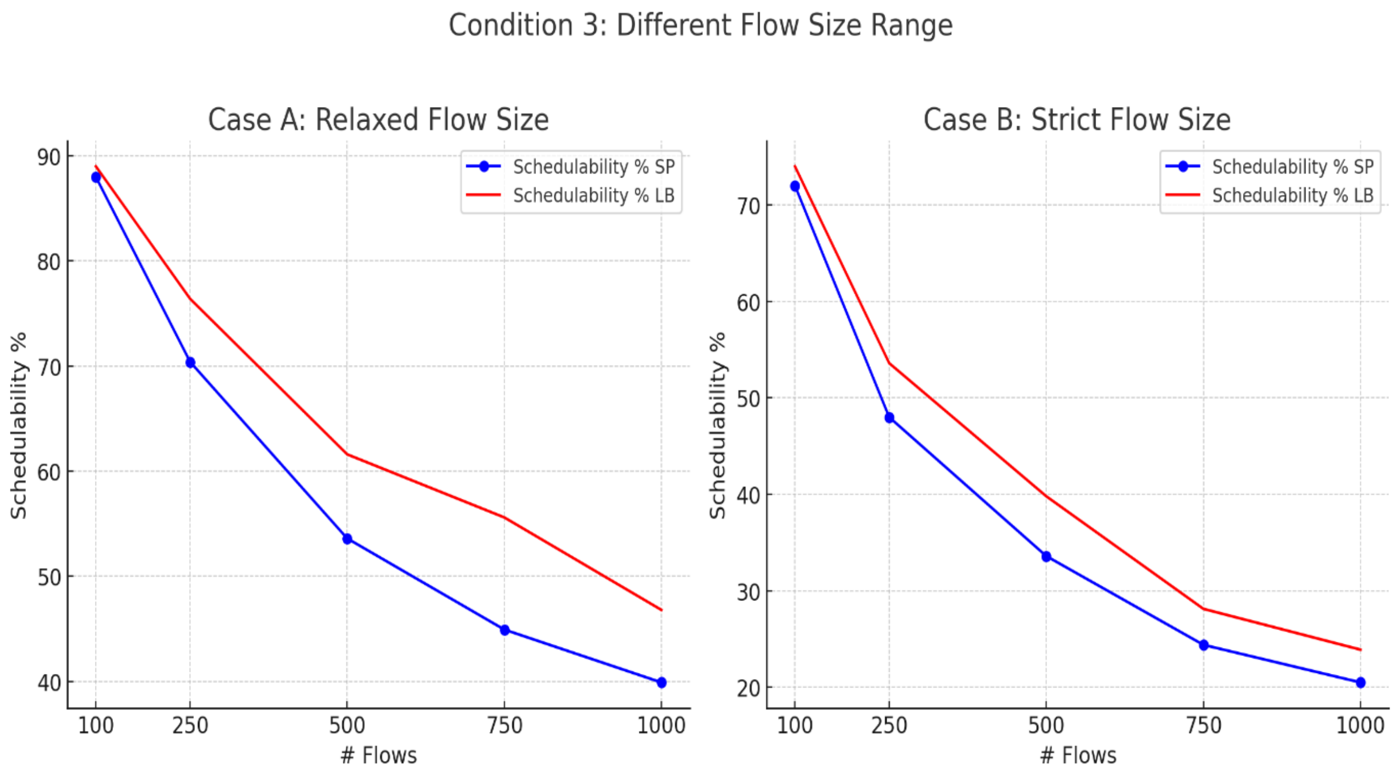


Results: Condition 3

Other Constant Values:

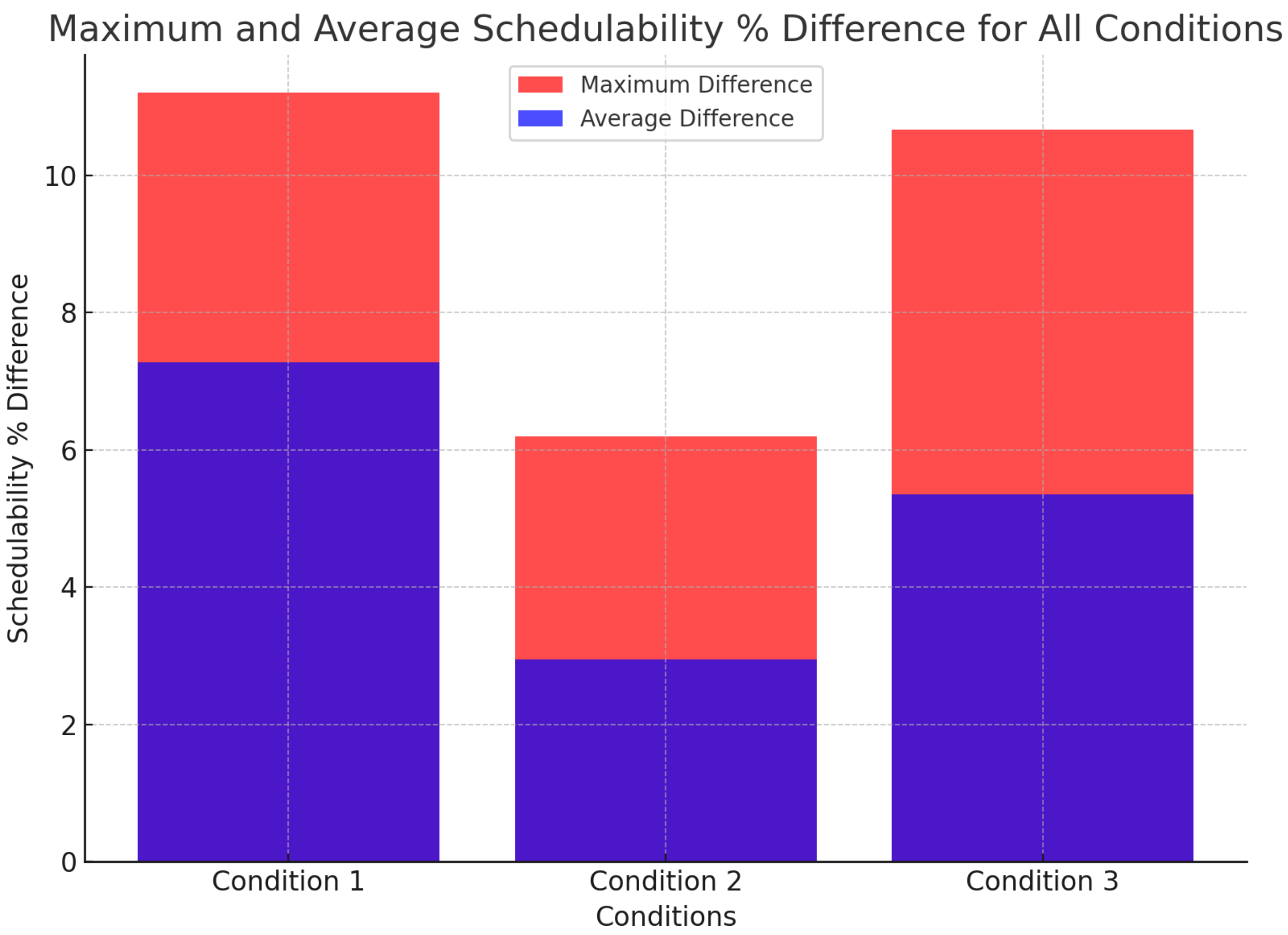
1. Sorting of flows: Shortest Deadline and period first

2. Deadline and Period=[1000, 2000, 5000, 10000]



Results:

Schedulability percentage difference



Summary and Future work:

- Algorithm used in my theses performed better.
- Sorting technique also has an impact.
- Different BW for bottleneck links.
- Algorithm can be used with combination of different shapers.

Thank you

- Questions?