NSF 1815866 III-Small

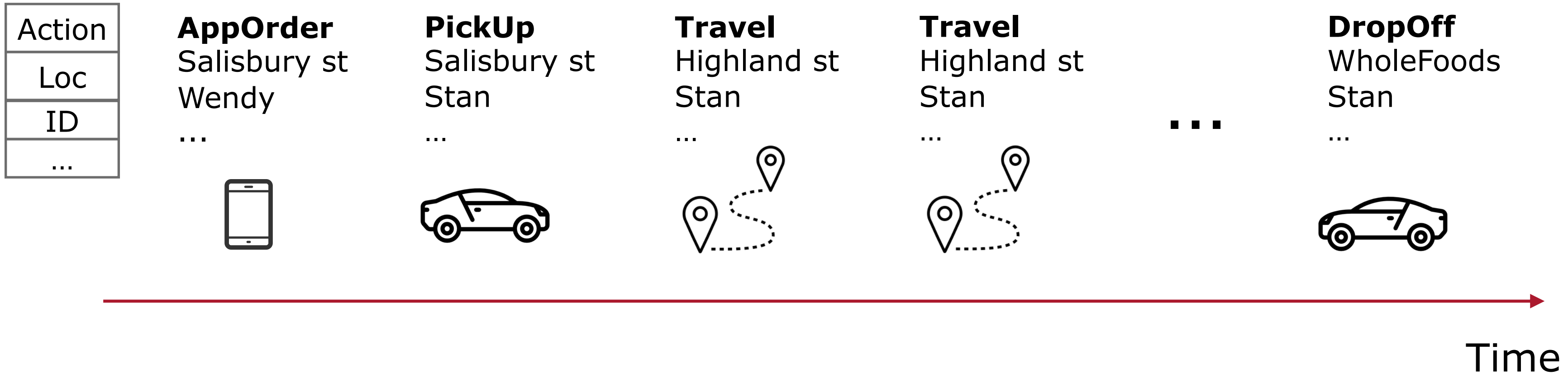
Gloria: Graph-based Sharing Optimizer for Event Trend Aggregation

Lei Ma, Chuan Lei, Olga Poppe and Elke Rundensteiner.

SIGMOD 2022

Motivation

Pattern and Event Trend

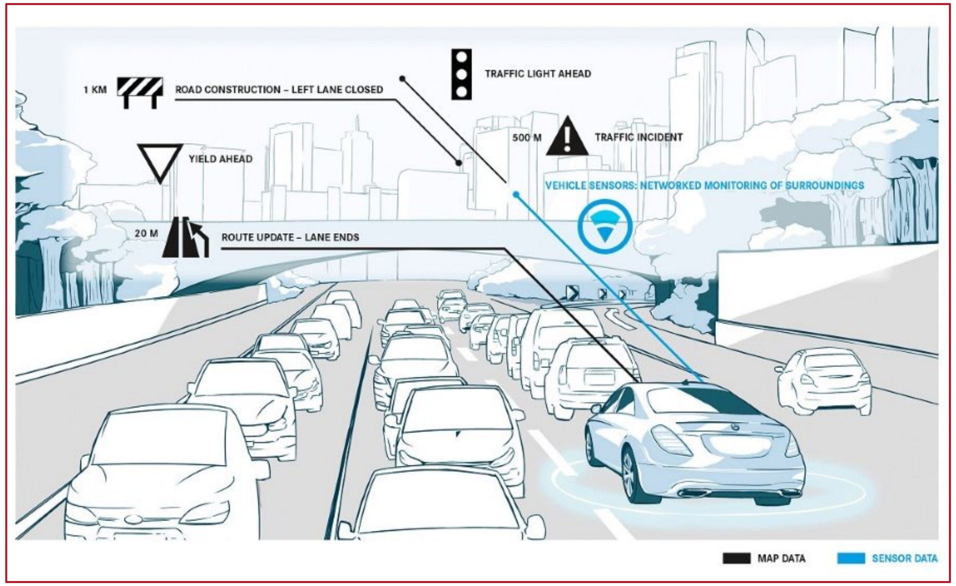


An **Event Trend (Complex Event)** can be defined by a **Pattern of Event types**.

e.g.
Event Trend - A **finished** trip
Pattern - SEQ(A, P, T+, D)

Event Trend Aggregation Queries

- How many trips finished in 30 minutes?
- How many trips get cancelled in 30 minutes?



Event Trend Aggregation Queries

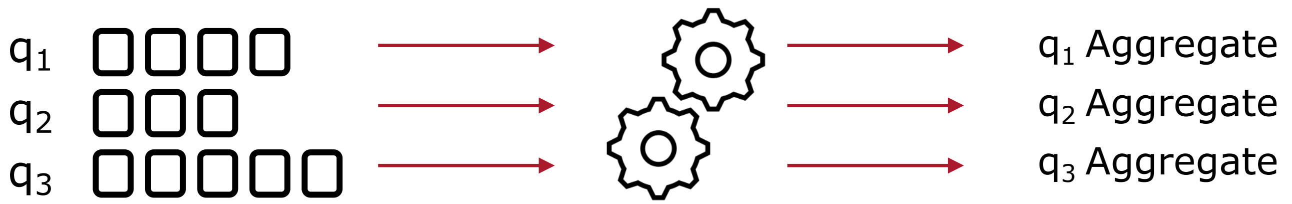
- How many trips finished in 30 minutes?

RETURN *COUNT(*)* → Aggregation Function
PATTERN *SEQ(A, P, T+, D)* → Pattern
WHERE *[driver_id]* **GROUP-BY** *[district]* → Predicates, Group-by
WINDOW *30 min* **SLIDE** *5 min* → Window of the aggregation

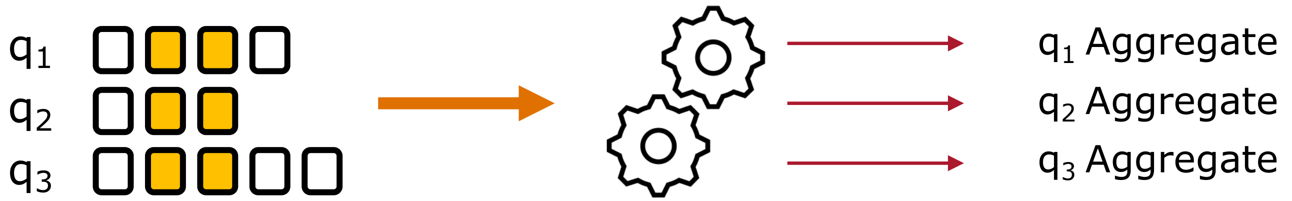
- How many trips get cancelled in 30 minutes?

RETURN *COUNT(*)*
PATTERN *SEQ(A, P, T+, C)*
WHERE *[driver_id]* **GROUP-BY** *[district]*
WINDOW *30 min* **SLIDE** *5 min*

Pattern Sharing

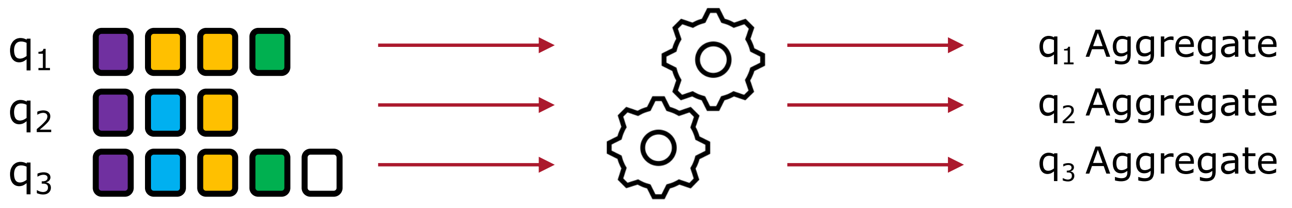


Execution without pattern sharing

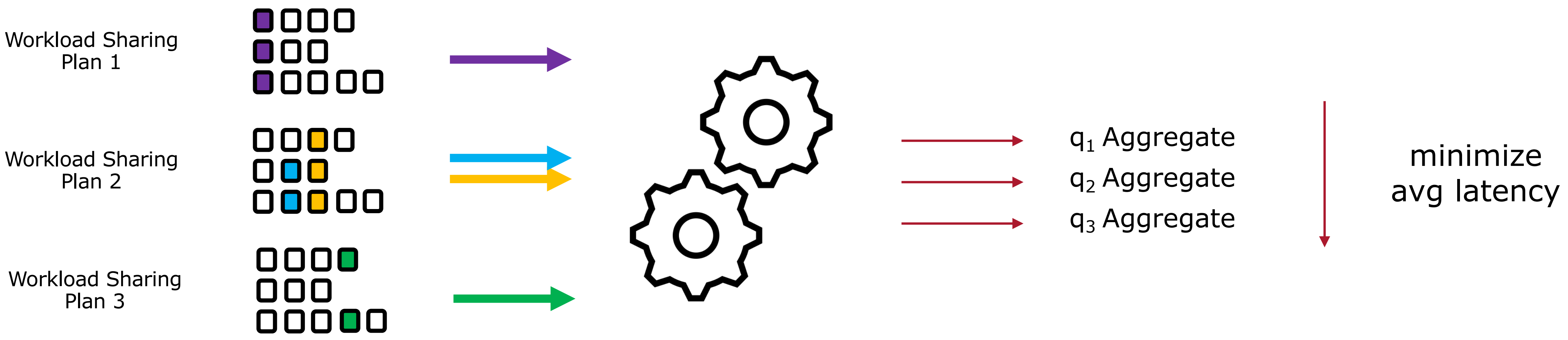


Execution with pattern sharing

Pattern Sharing



Execution with pattern sharing



Challenges

1. Query Complexity

Kleene operators in the patterns

e.g. SEQ(A, P, T+, C)

Kleene patterns could be Nested

e.g. SEQ(A, P, (T+, C)+)

→

1. An Exponential number of matched Event Trends

2. Hard to analyze the sharing opportunities

Challenges

1. Query Complexity
2. Sharing Complexity

Analyze sharing opportunities →

1. Flexibility of sharing
2. An accurate cost model

Challenges

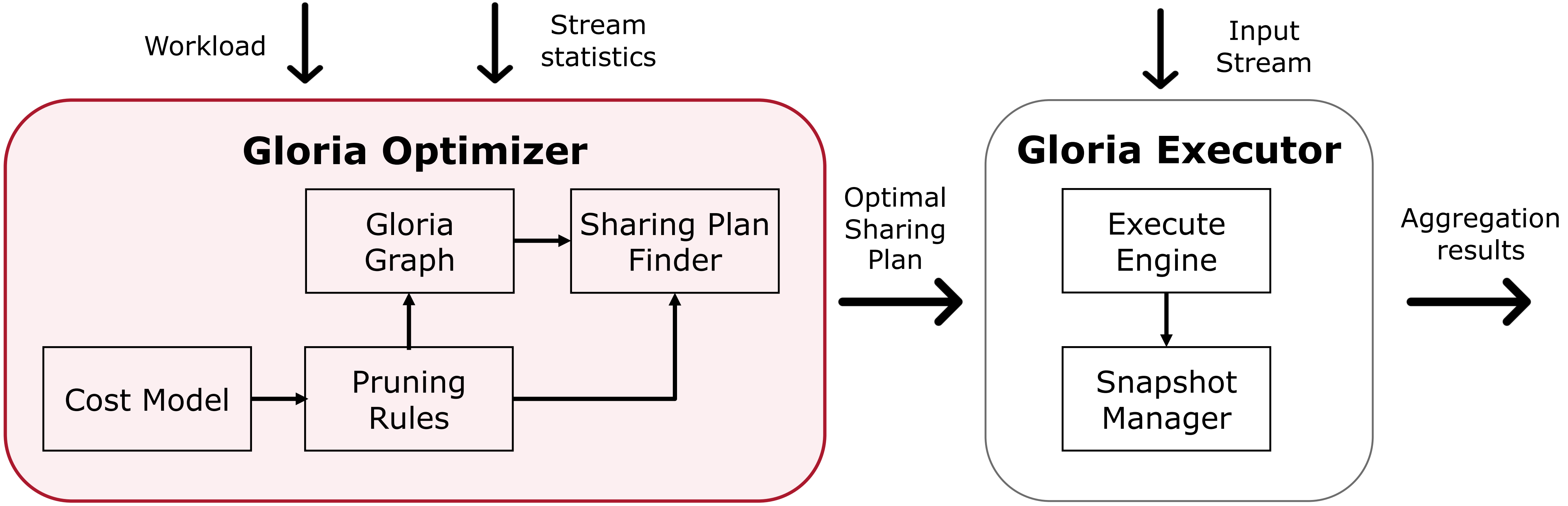
1. Query Complexity
2. Sharing Complexity
3. Search Complexity

Flexibility of Arbitrary sharing → An exponential search space

State-of-the-art

Approach	Aggregation Strategy	Kleene pattern type	Sharing decisions
MCEP [SIGMOD'19]	Two-step	Restricted	Flexible
Sharon [ICDE'18]	Online	-	Restricted
Greta [VLDB'17]	Online	General	-
Hamlet [SIGMOD'21]	Online	Restricted	Restricted
GLORIA[SIGMOD'22]	**Online**	**General**	**Flexible**

Gloria Framework

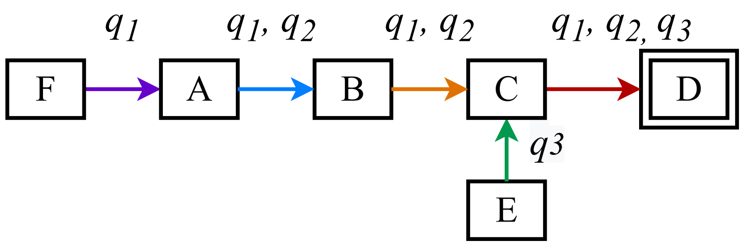


Gloria Graph Model

Sharing Opportunities

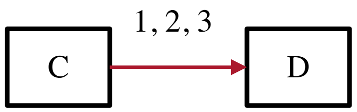
q_1: SEQ(F, A, B, C, D)
q_2: SEQ(A, B, C, D)
q_3: SEQ(E, C, D)

Workload



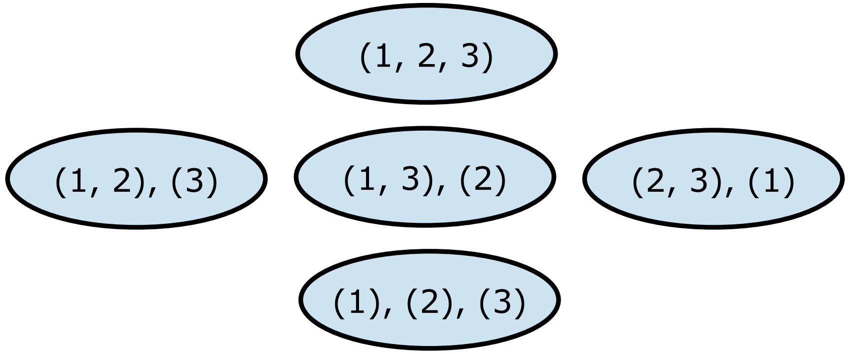
Template

Transition Sharing Plan



Transition (C, D)

shared groups



k =1

k =2

k =3

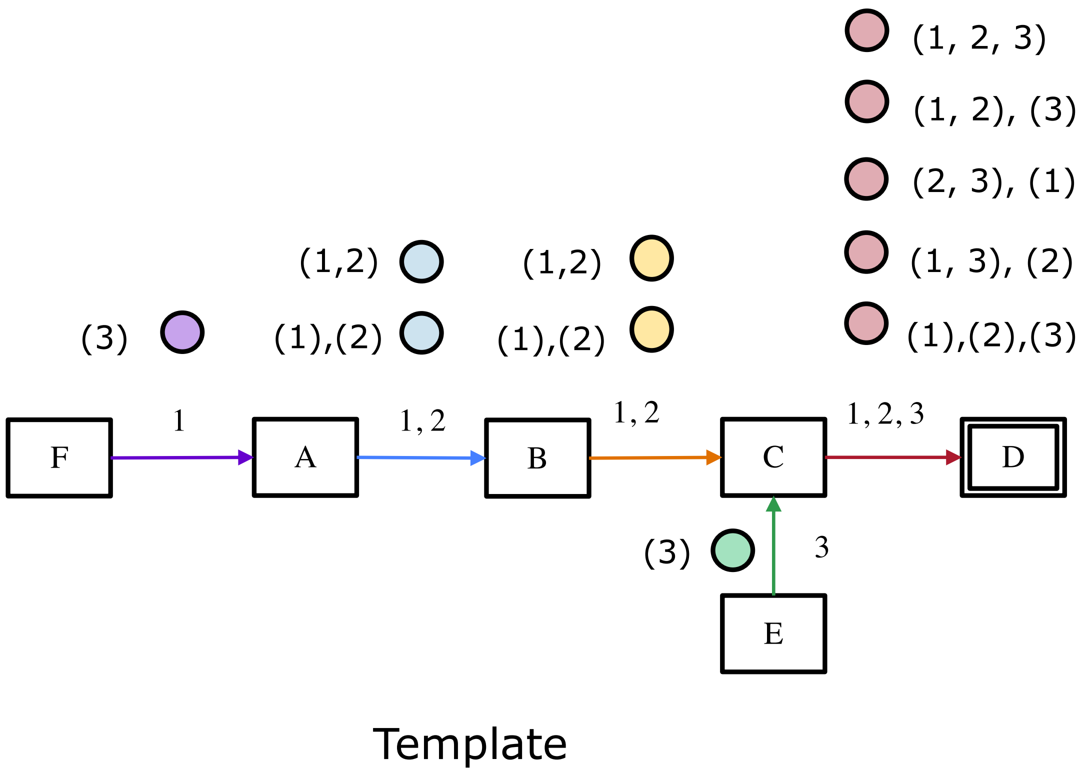
5 different transition sharing plans

$$B_n = e^{e^n}$$

n- number of queries

Workload Sharing Plan

- A combination of transition sharing plans for all transitions.



Transition sharing plan

$$B_n^m = e^{e^n * m}$$

n- number of queries

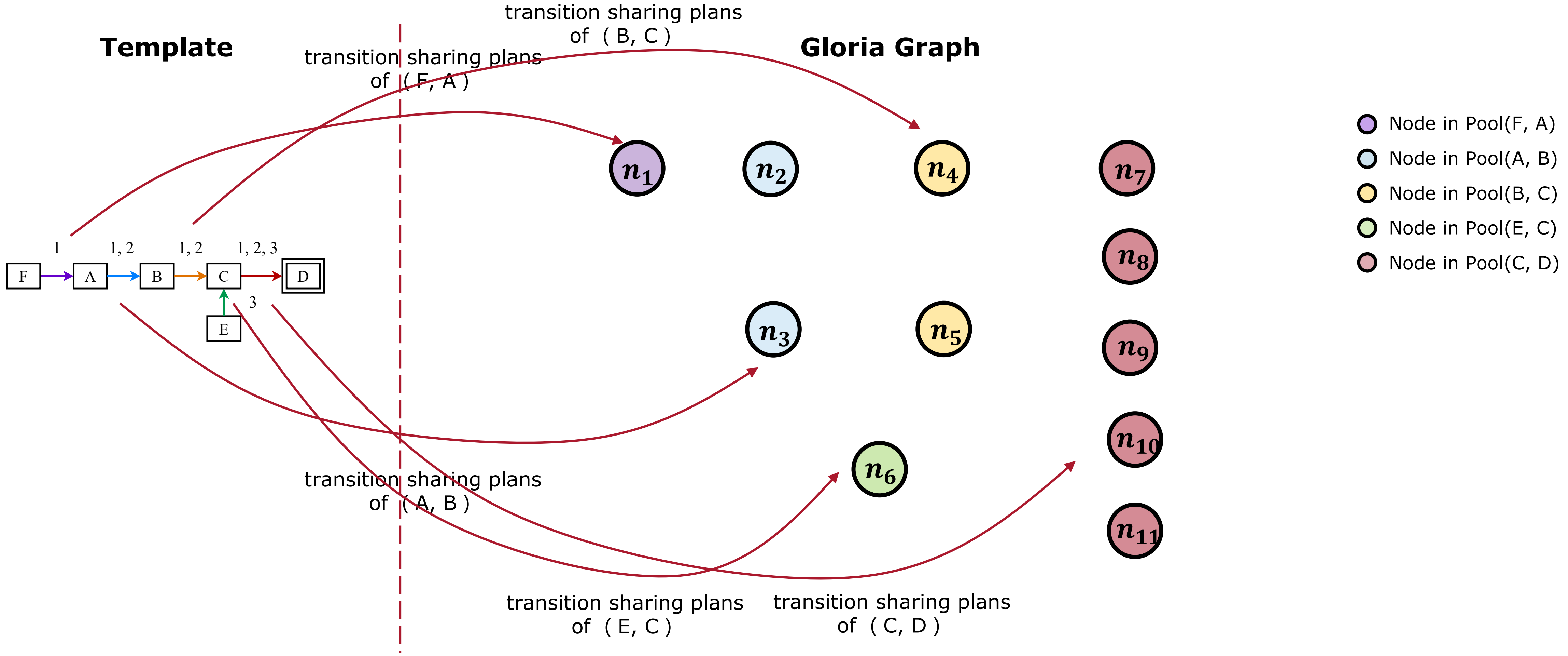
m- number of transitions

Problem Transformation

Search space → Gloria Graph

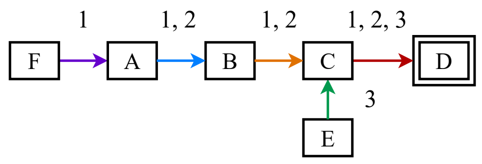
Optimal workload sharing plan → Optimal path with minimum weight

Gloria Graph Model

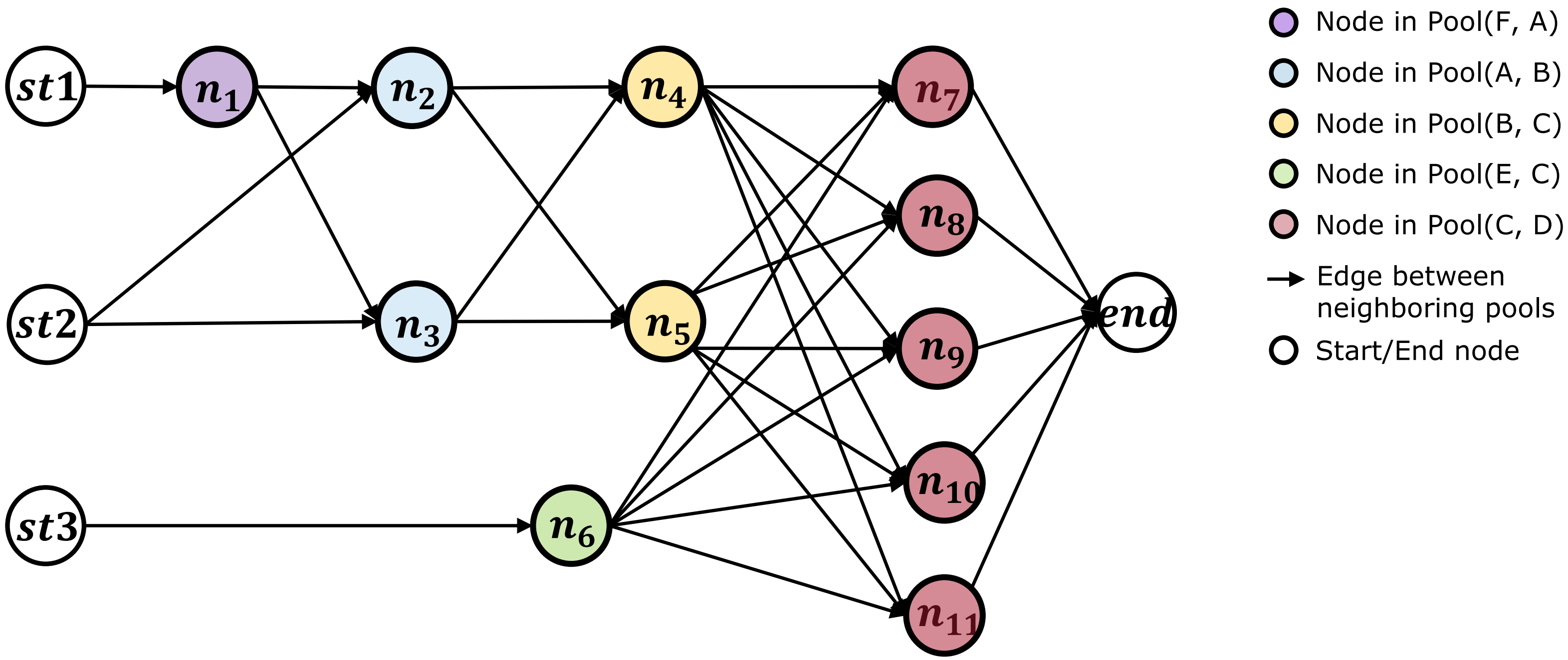


Gloria Graph Model

Template

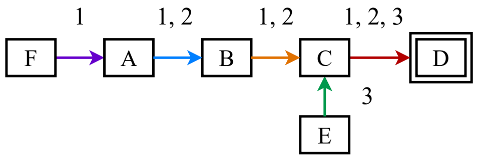


Gloria Graph

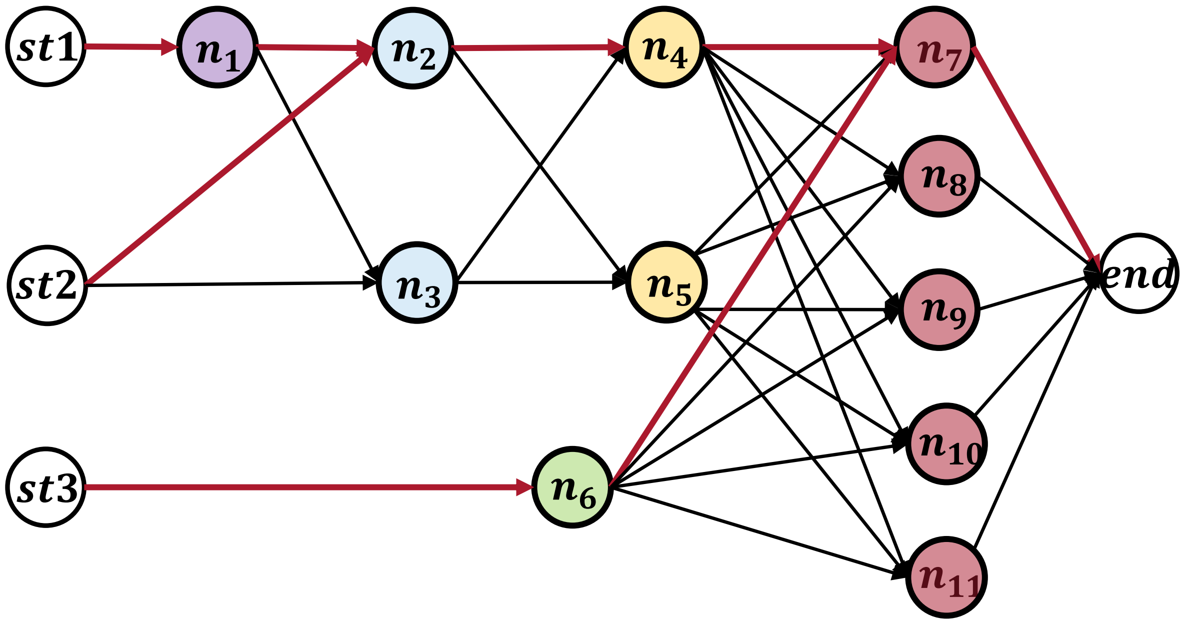


Gloria Graph Model

Template



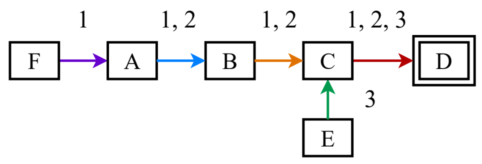
Gloria Graph



Node in Pool(F, A)
Node in Pool(A, B)
Node in Pool(B, C)
Node in Pool(E, C)
Node in Pool(C, D)
Edge between neighboring pools
Start/End node
A path

Gloria Graph Model

Template

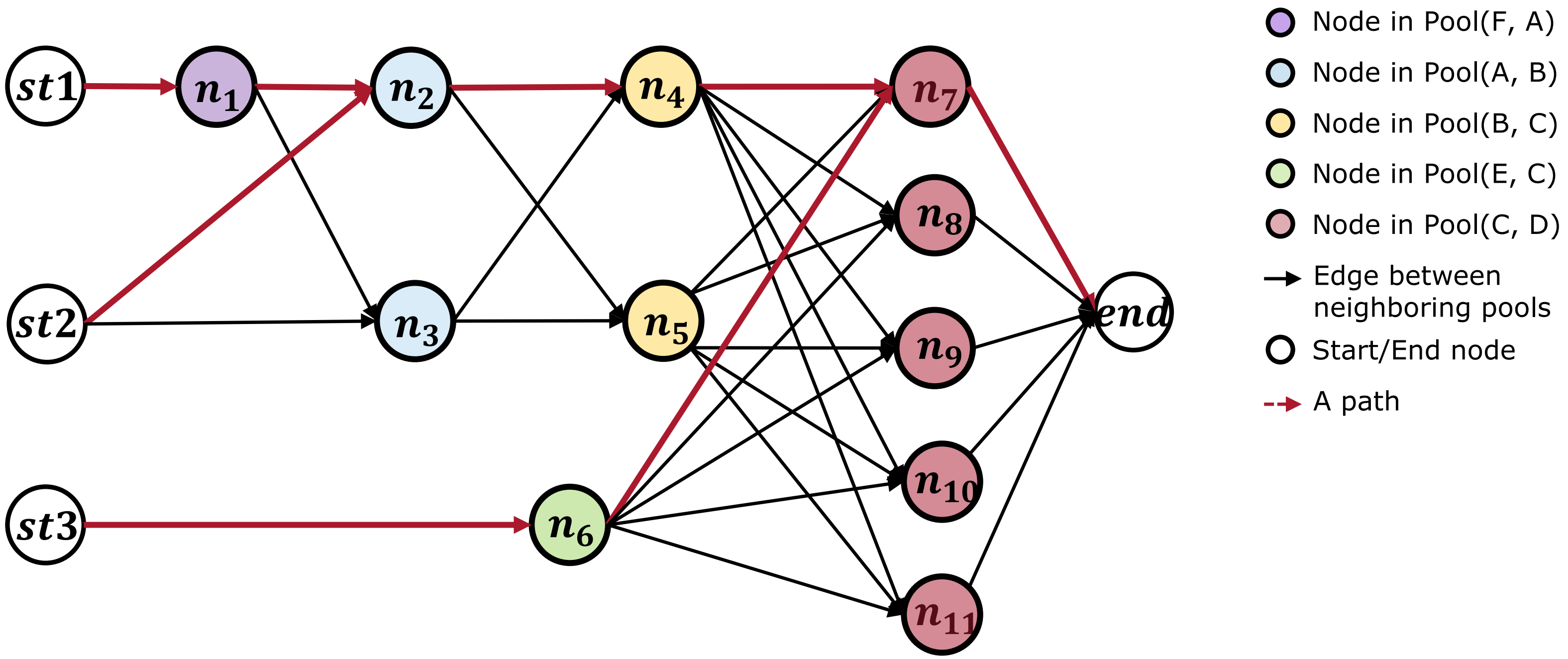


The total execution cost of a workload sharing plan is:

$$Cost(Q|\Omega) = \sum_i Cost(E_i|\Omega)$$

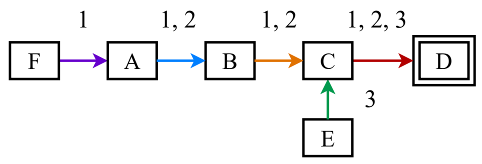
Q – workload
Ω – a workload sharing plan
E_i – an event type

Gloria Graph



Gloria Graph Model

Template

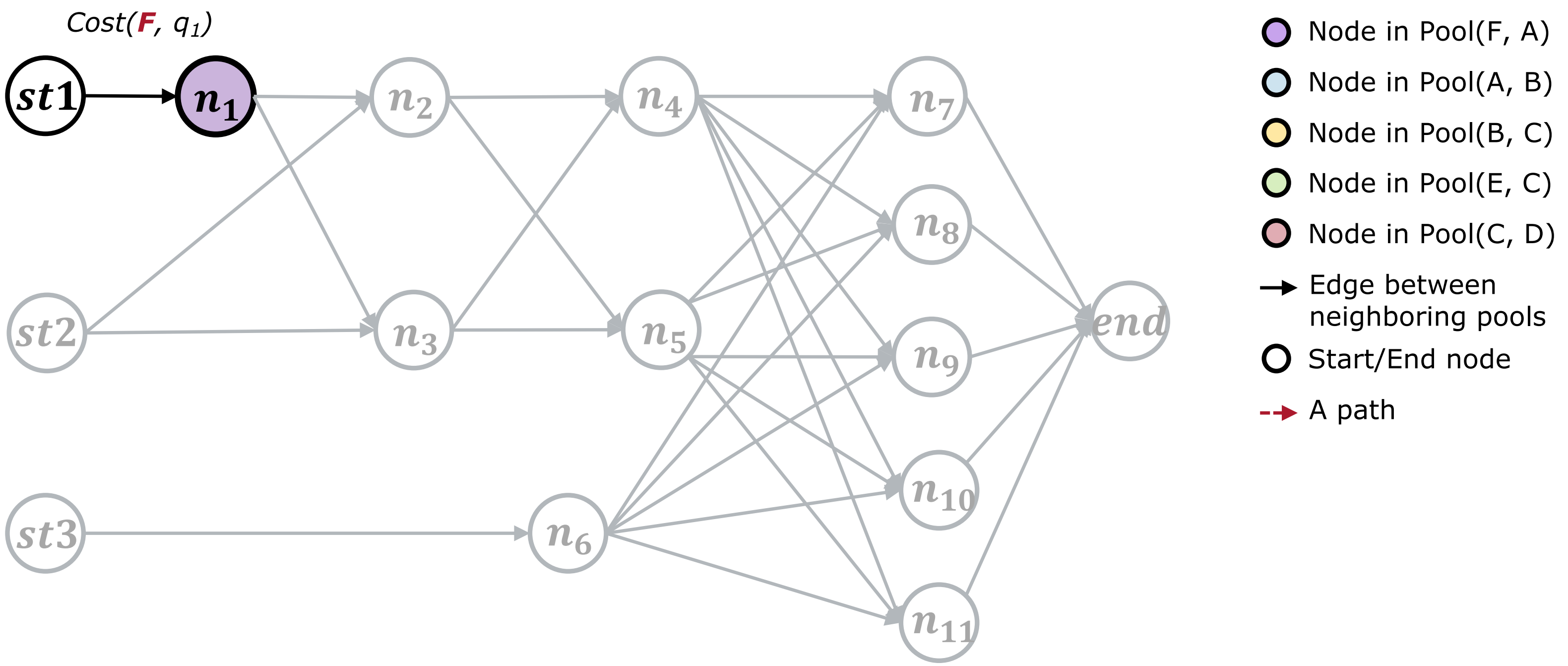


The total execution cost of a workload sharing plan is:

$$Cost(Q|\Omega) = \sum_i Cost(E_i|\Omega)$$

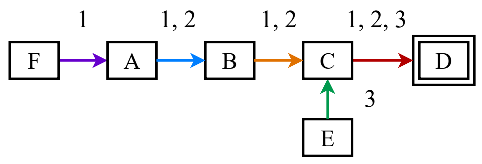
Q – workload
Ω – a workload sharing plan
E_i – an event type

Gloria Graph



Gloria Graph Model

Template

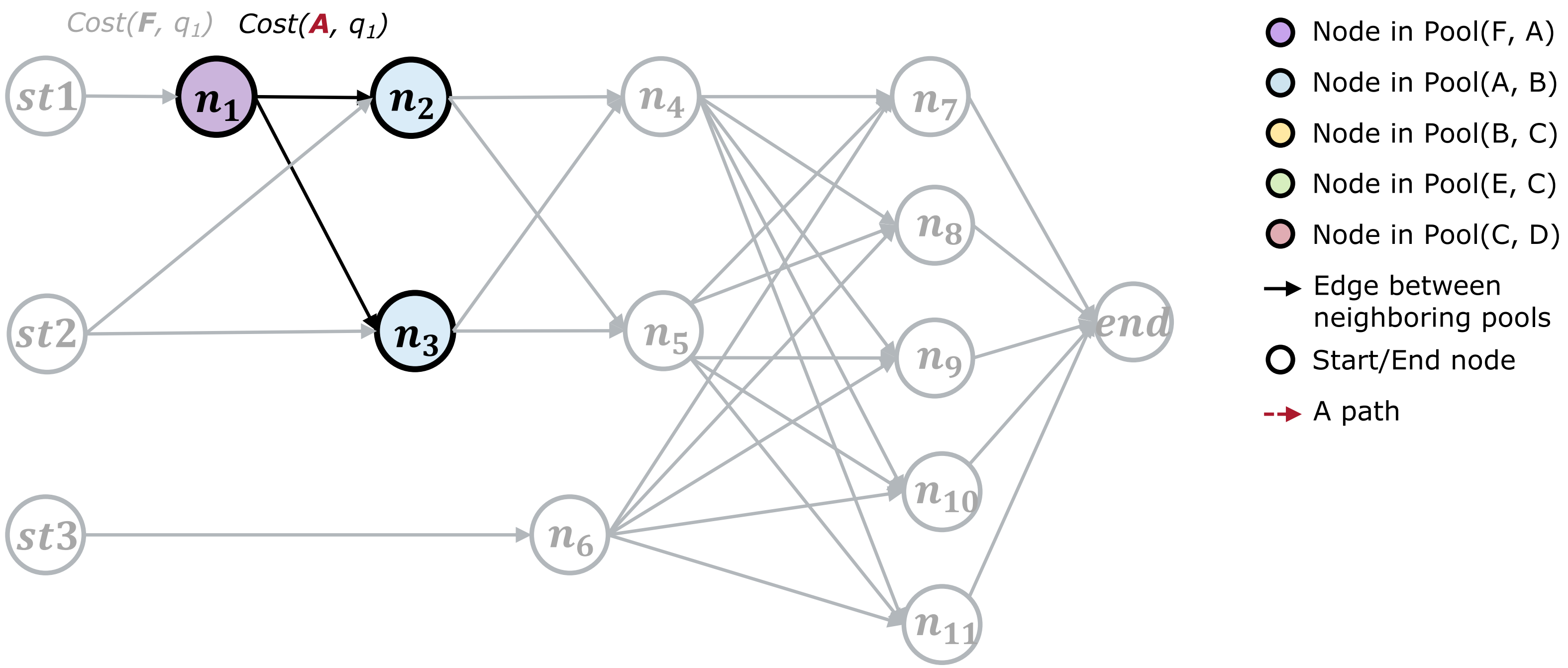


The total execution cost of a workload sharing plan is:

$$Cost(Q|\Omega) = \sum_i Cost(E_i|\Omega)$$

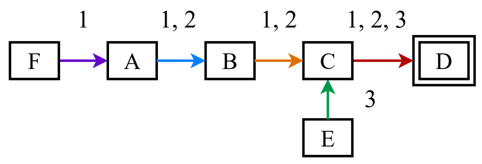
Q – workload
Ω – a workload sharing plan
E_i – an event type

Gloria Graph



Gloria Graph Model

Template

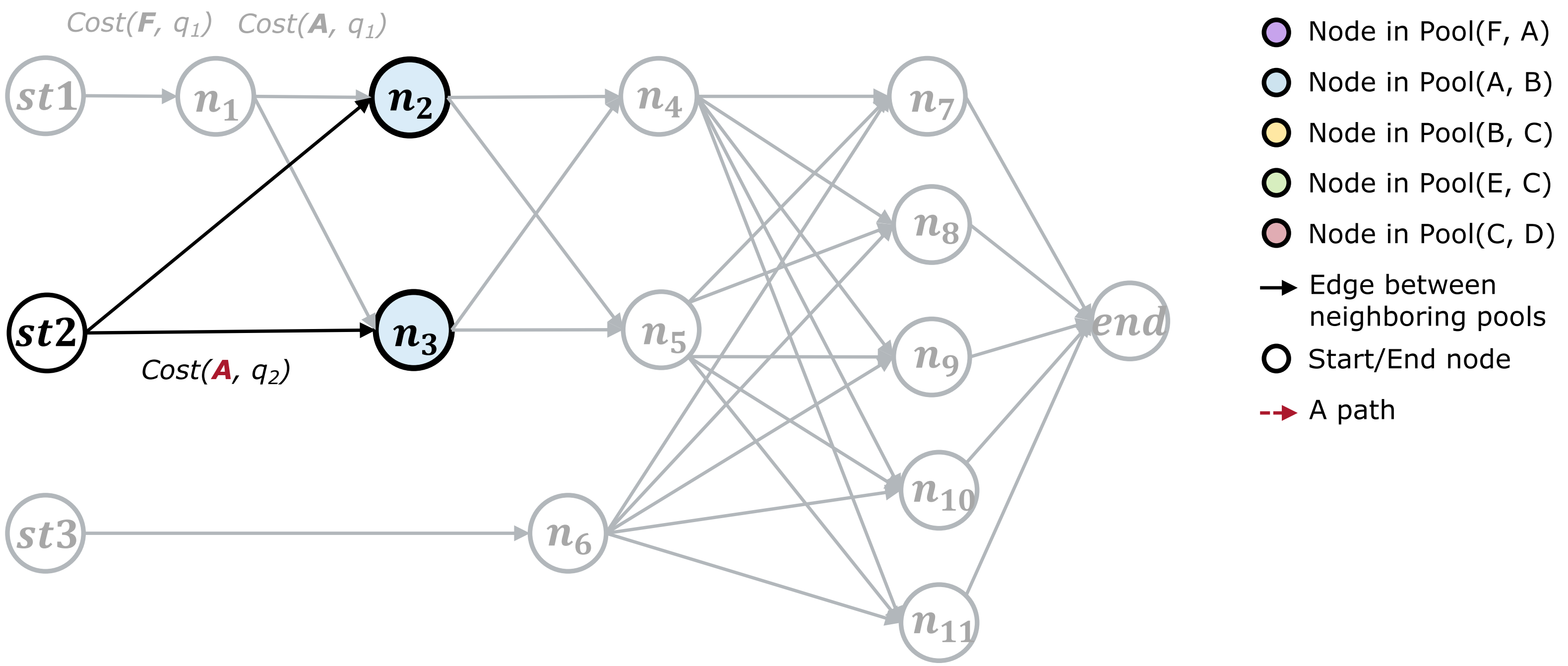


The total execution cost of a workload sharing plan is:

$$Cost(Q|\Omega) = \sum_i Cost(E_i|\Omega)$$

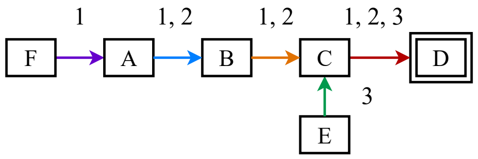
Q – workload
Ω – a workload sharing plan
E_i – an event type

Gloria Graph



Gloria Graph Model

Template

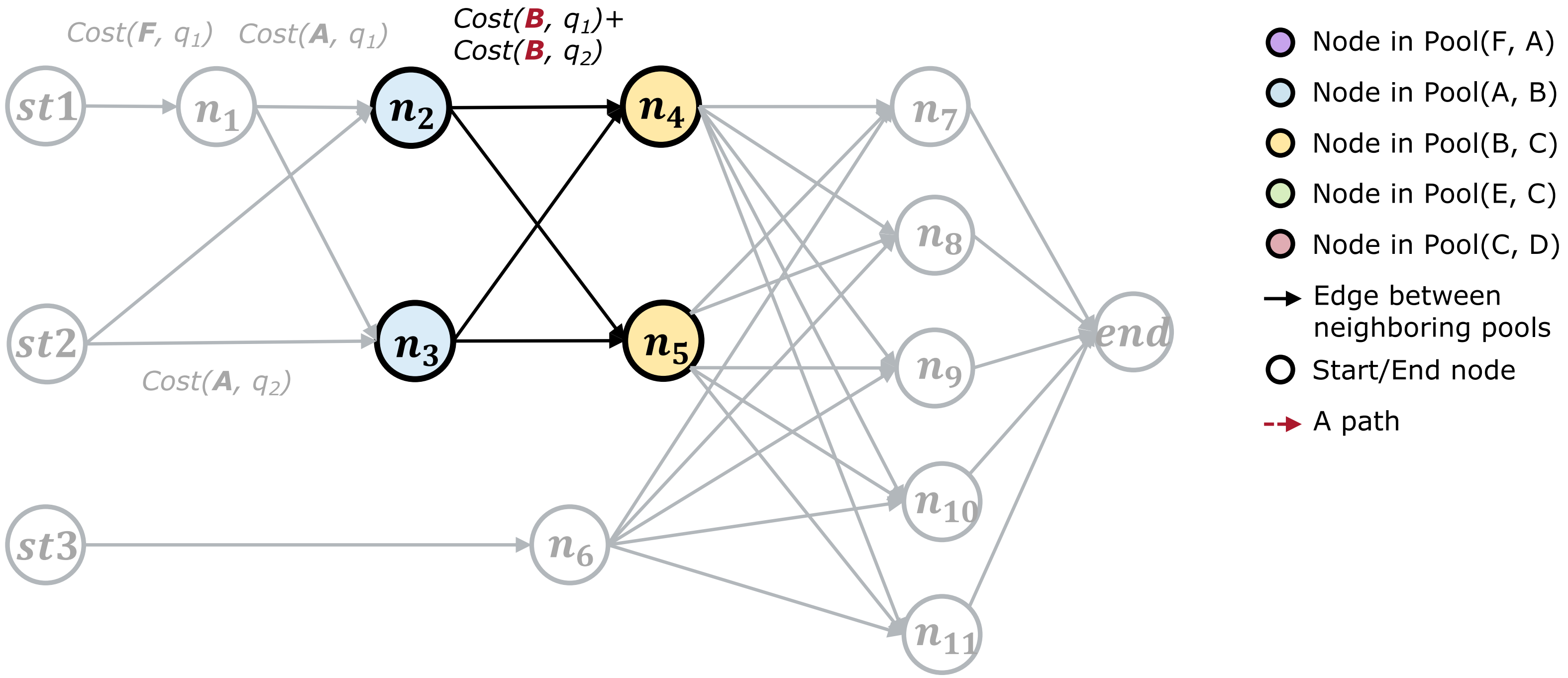


The total execution cost of a workload sharing plan is:

$$Cost(Q|\Omega) = \sum_i Cost(E_i|\Omega)$$

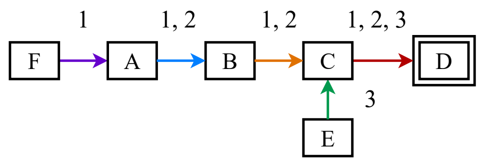
Q – workload
Ω – a workload sharing plan
E_i – an event type

Gloria Graph



Gloria Graph Model

Template

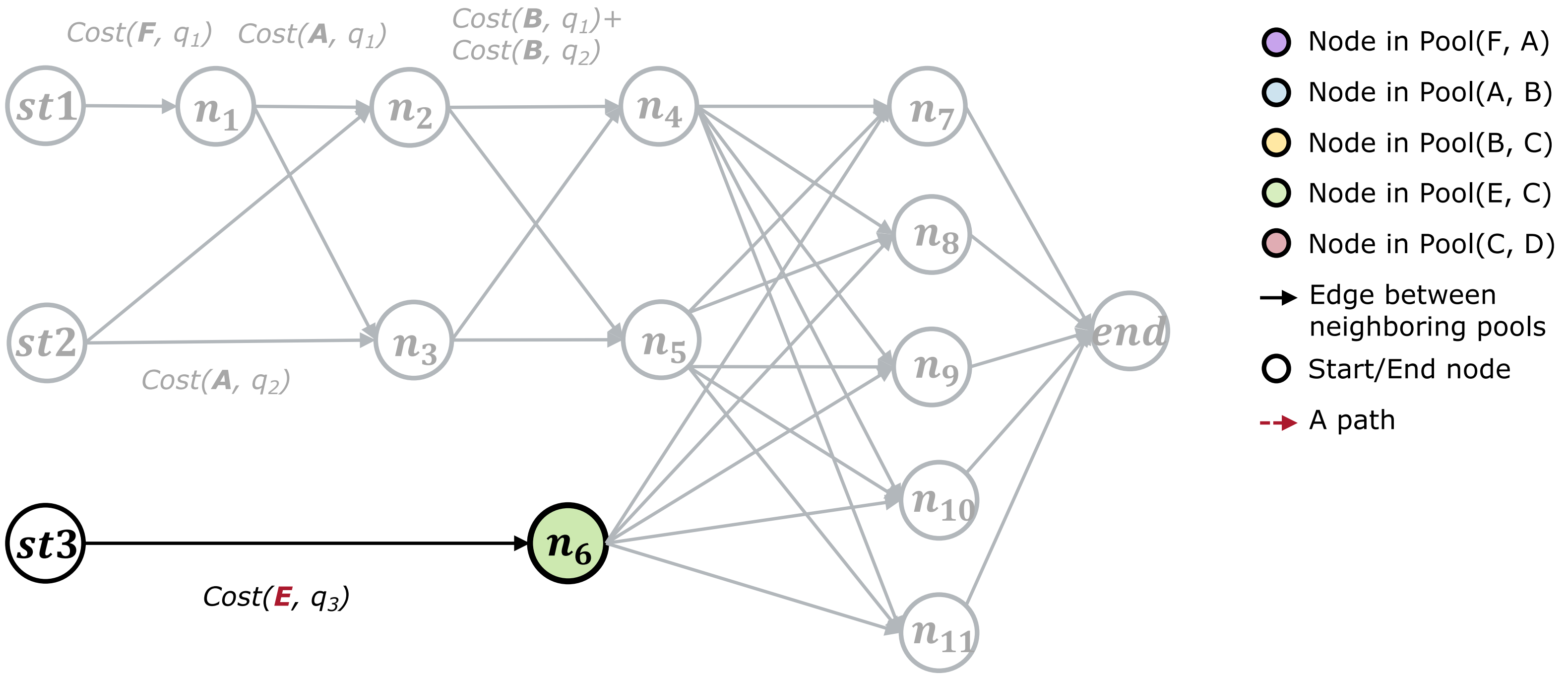


The total execution cost of a workload sharing plan is:

$$Cost(Q|\Omega) = \sum_i Cost(E_i|\Omega)$$

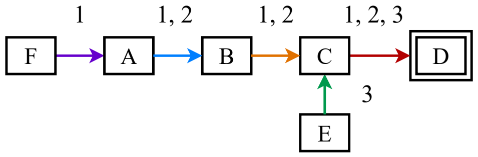
Q – workload
Ω – a workload sharing plan
E_i – an event type

Gloria Graph



Gloria Graph Model

Template

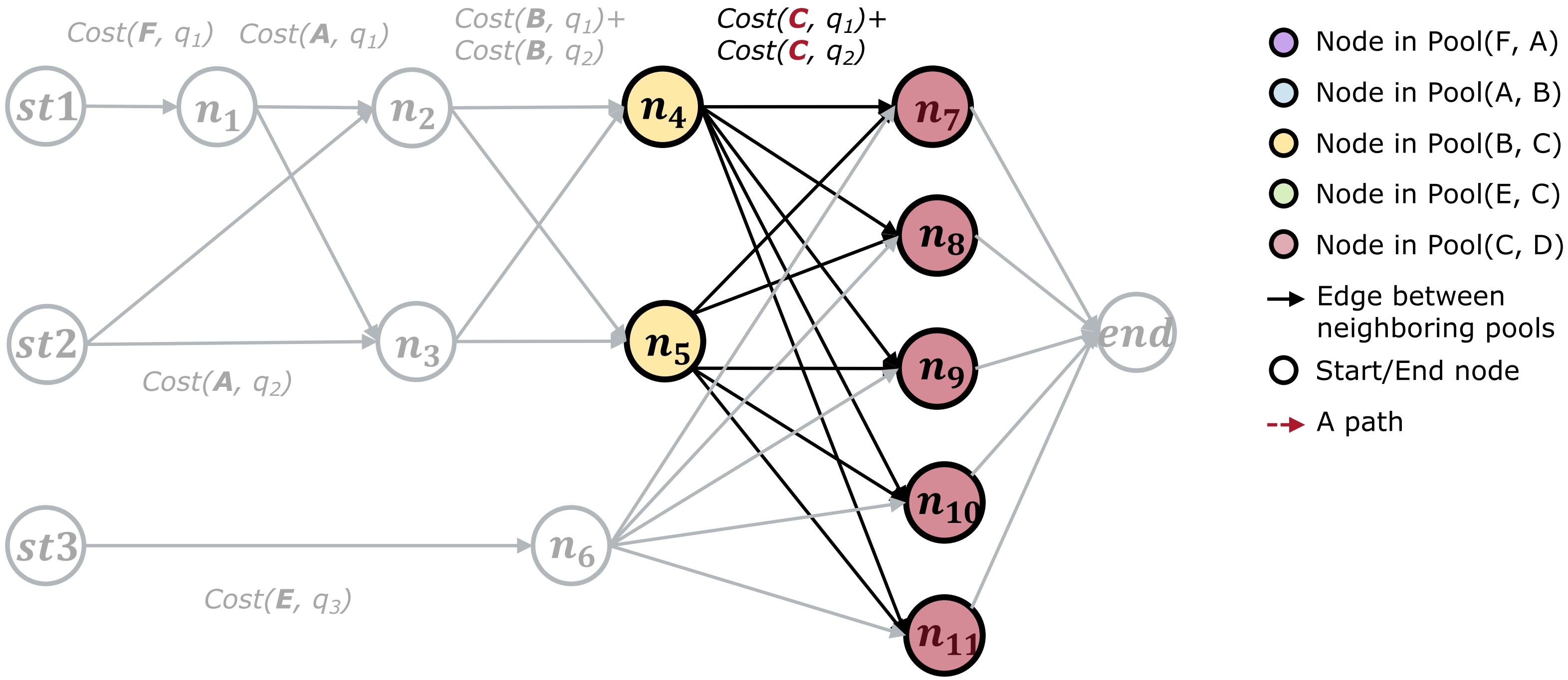


The total execution cost of a workload sharing plan is:

$$Cost(Q|\Omega) = \sum_{i} Cost(E_i|\Omega)$$

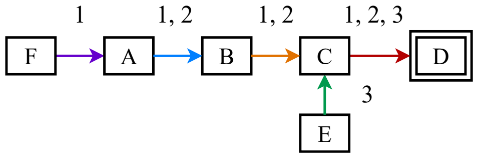
Q – workload
Ω – a workload sharing plan
E_i – an event type

Gloria Graph



Gloria Graph Model

Template

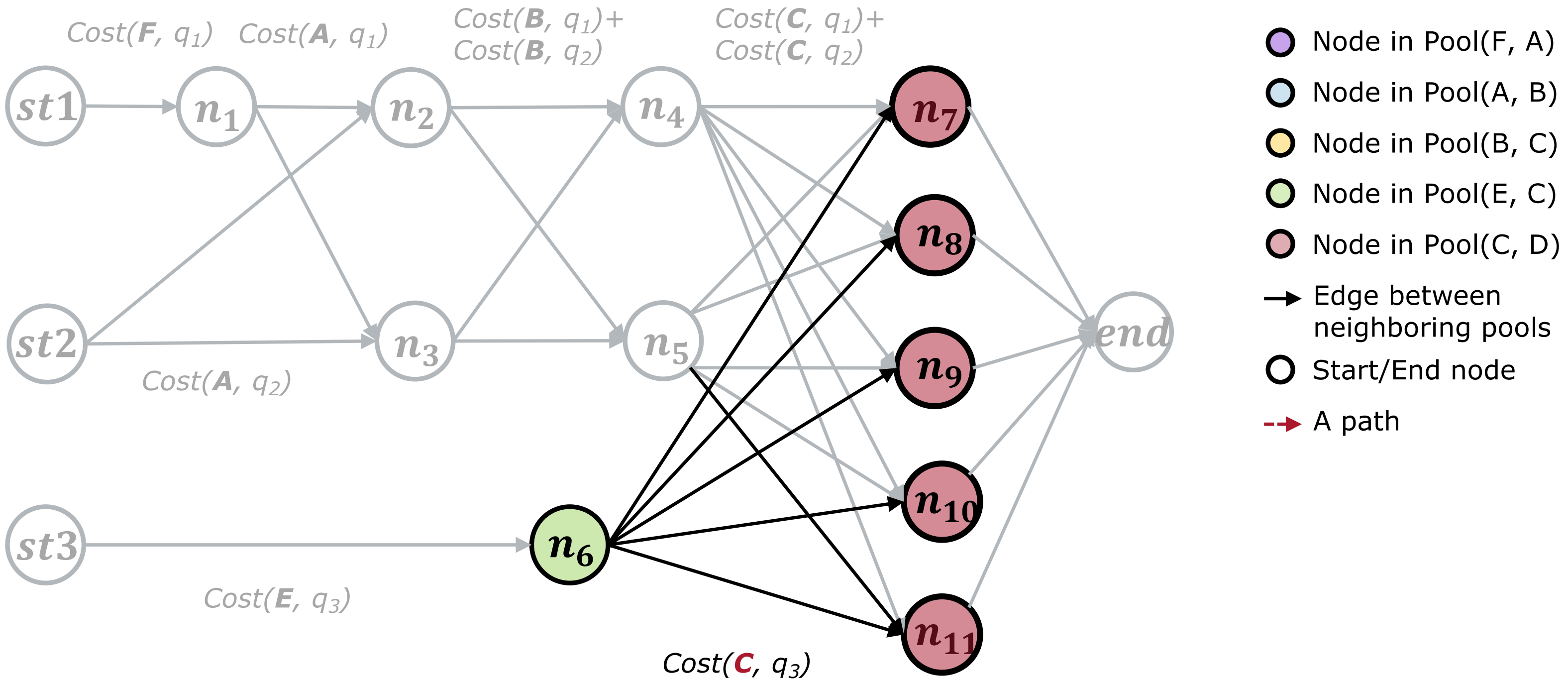


The total execution cost of a workload sharing plan is:

$$Cost(Q|\Omega) = \sum_{i} Cost(E_i|\Omega)$$

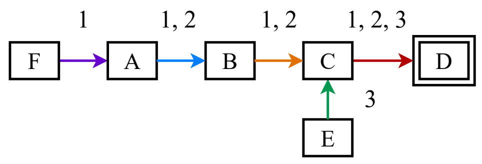
Q – workload
Ω – a workload sharing plan
E_i – an event type

Gloria Graph



Gloria Graph Model

Template

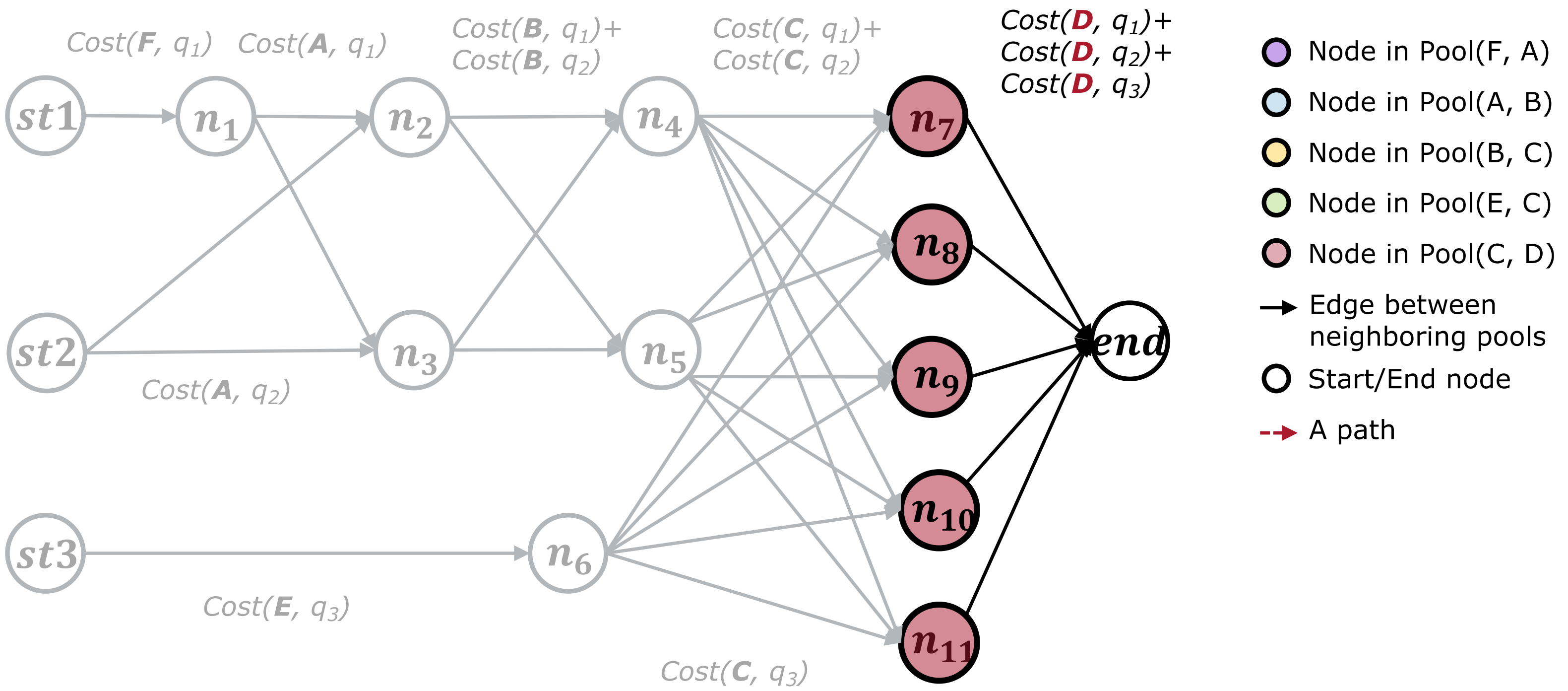


The total execution cost of a workload sharing plan is:

$$Cost(Q|\Omega) = \sum_i Cost(E_i|\Omega)$$

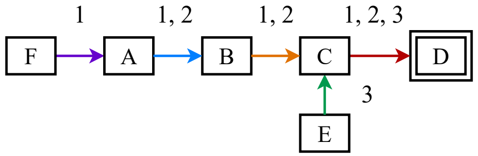
Q – workload
Ω – a workload sharing plan
E_i – an event type

Gloria Graph



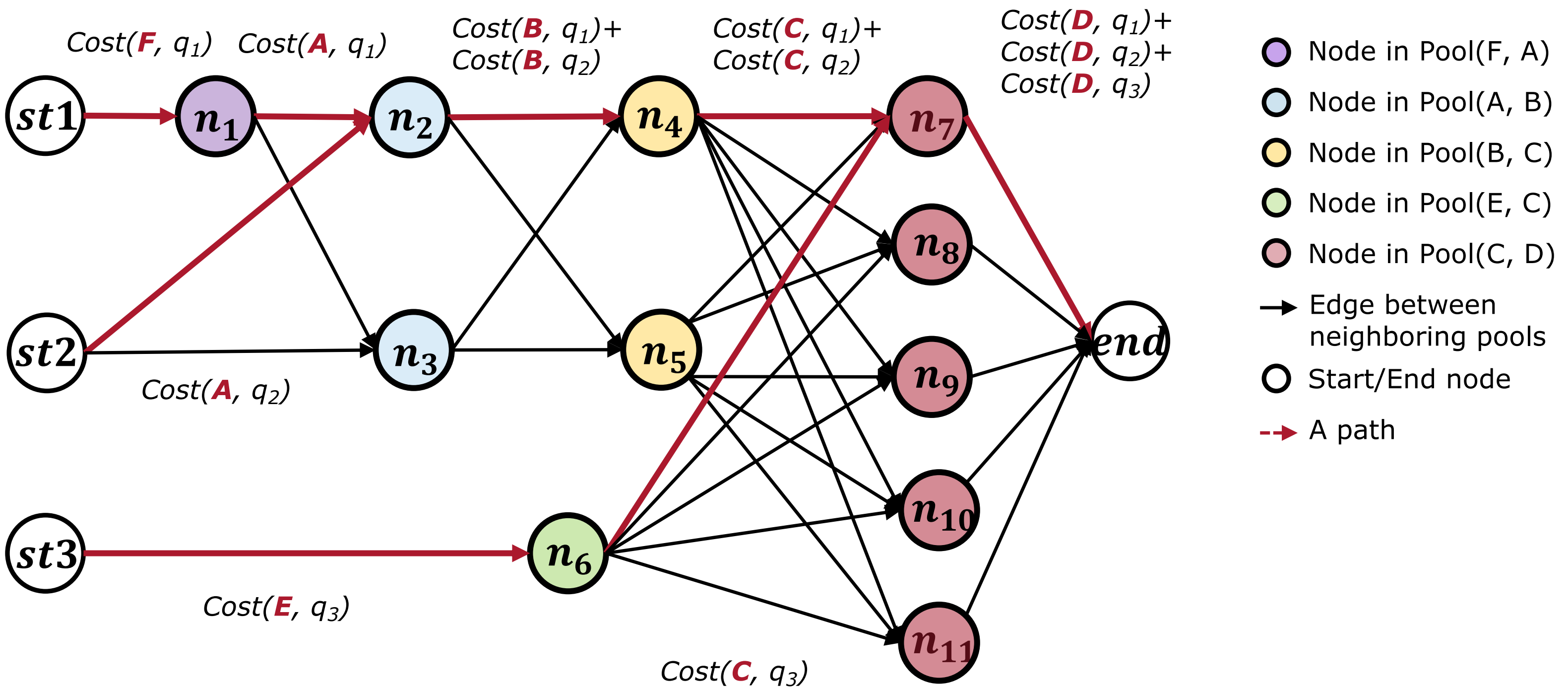
Gloria Graph Model

Template



Q – workload
Ω – a workload sharing plan
$\widehat{\Omega}$ – optimal workload sharing plan

Gloria Graph

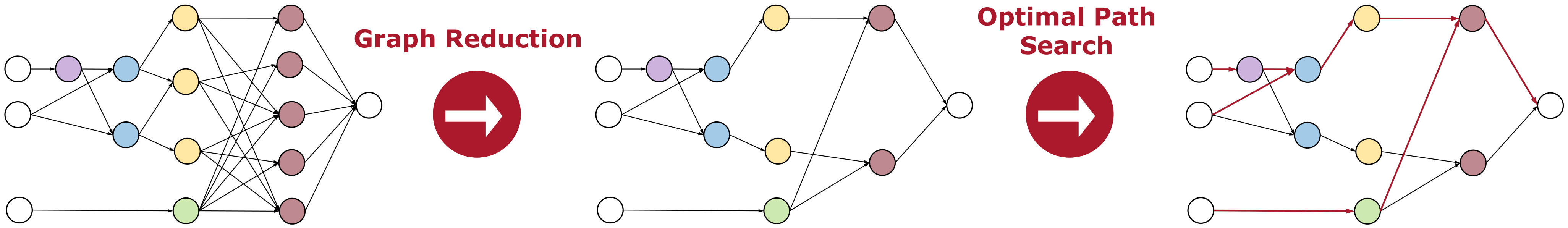


$$Cost(Q|\Omega) = \sum_i Cost(E_i|\Omega) \quad \longleftrightarrow \quad P.weight = \sum_i Cost(E_i|\Omega)$$

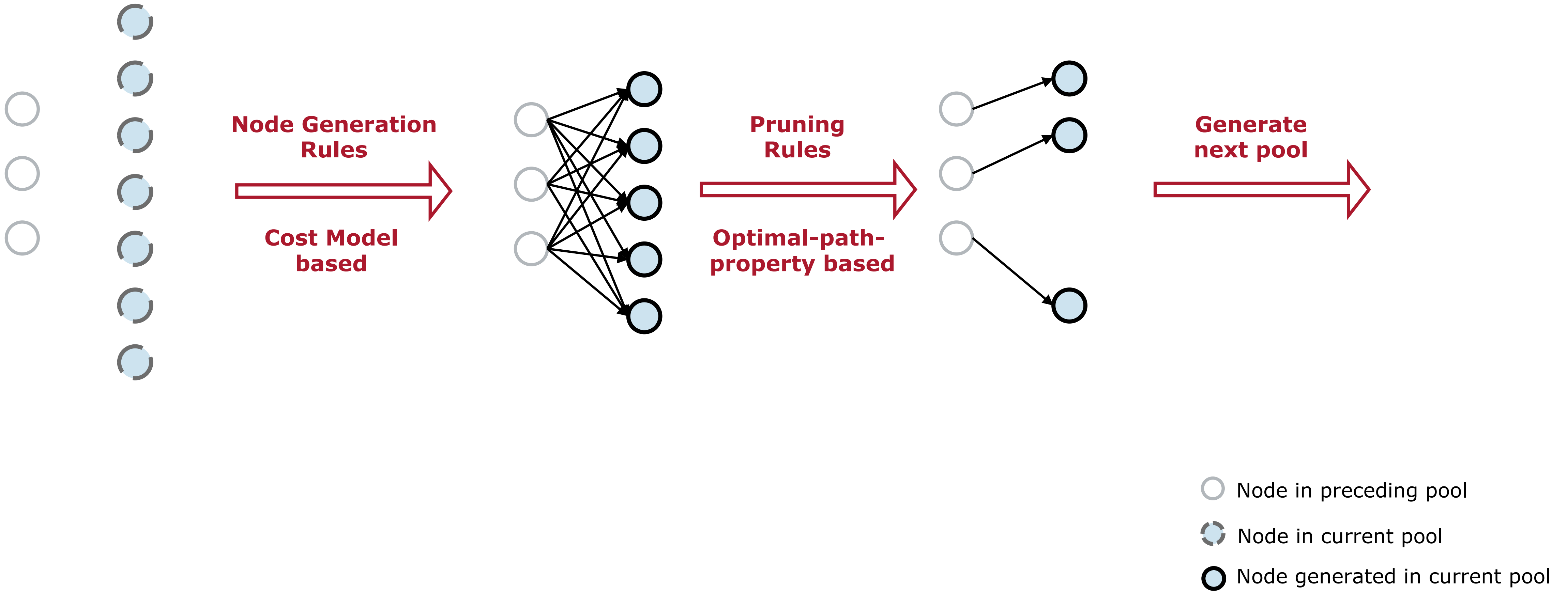
$$\widehat{\Omega} = \underset{\Omega}{\mathrm{Argmin}}\, Cost(Q|\Omega) \quad \longleftrightarrow \quad \widehat{P} = \underset{P}{\mathrm{Argmin}}\, P.weight$$

Gloria Optimizer

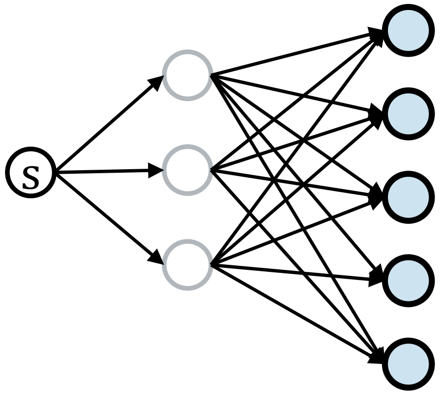
Optimization Goals



Graph Reduction



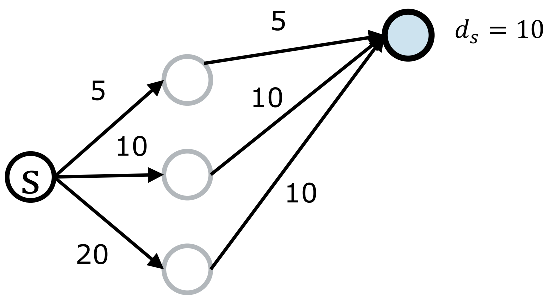
Pool Pruning-Edge



Node in preceding pool

Node in current pool

Pool Pruning-Edge



$path_1.weight = 10$

$path_2.weight = 20$

$path_3.weight = 30$

Node in preceding pool

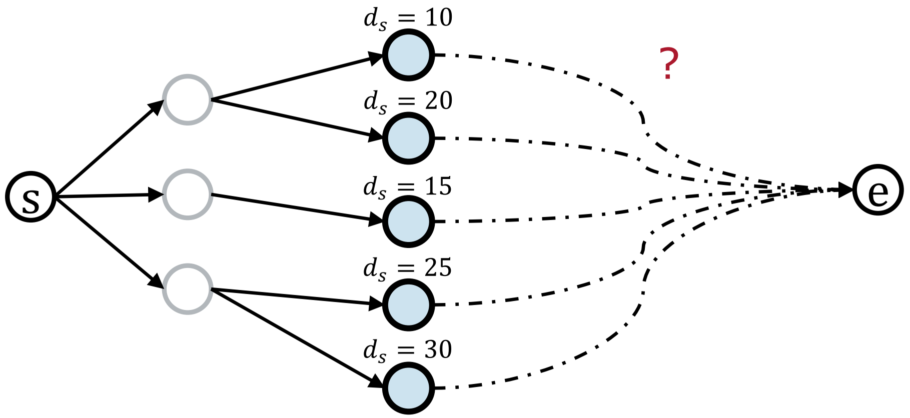
Node in current pool

Distance to the start node:

$$d_s = \min\{path_i.weight\} = 10$$

Each node only keep one incoming edge from a preceding pool

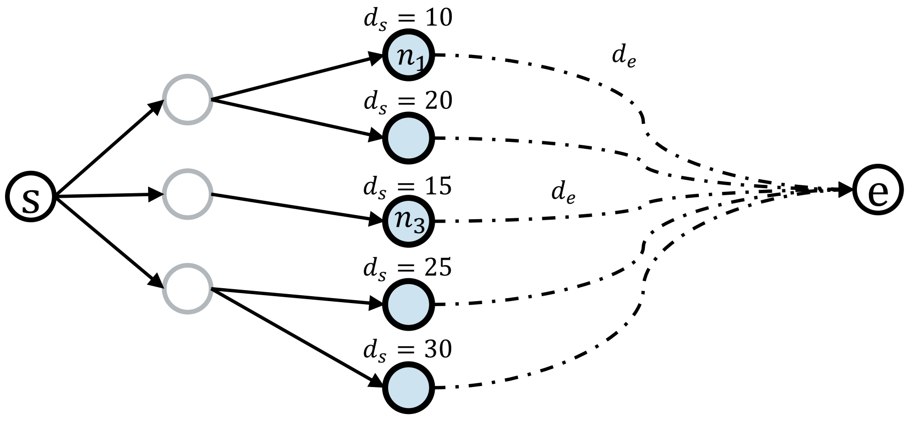
Pool Pruning-Node



But…
the distances to the end node are unknown.

Node in preceding pool
Node in current pool

Pool Pruning-Node



- Define comparable nodes

  n_1 and n_3 are comparable

- Estimate

  $n_1.d_e >$ or $< n_3.d_e$

- Prune

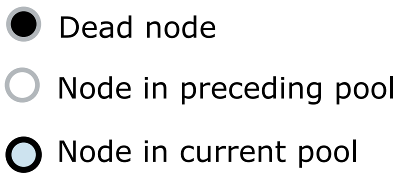
  $prune\ n_3\ if:$

  $n_1.d_s < n_3.d_s\ and$

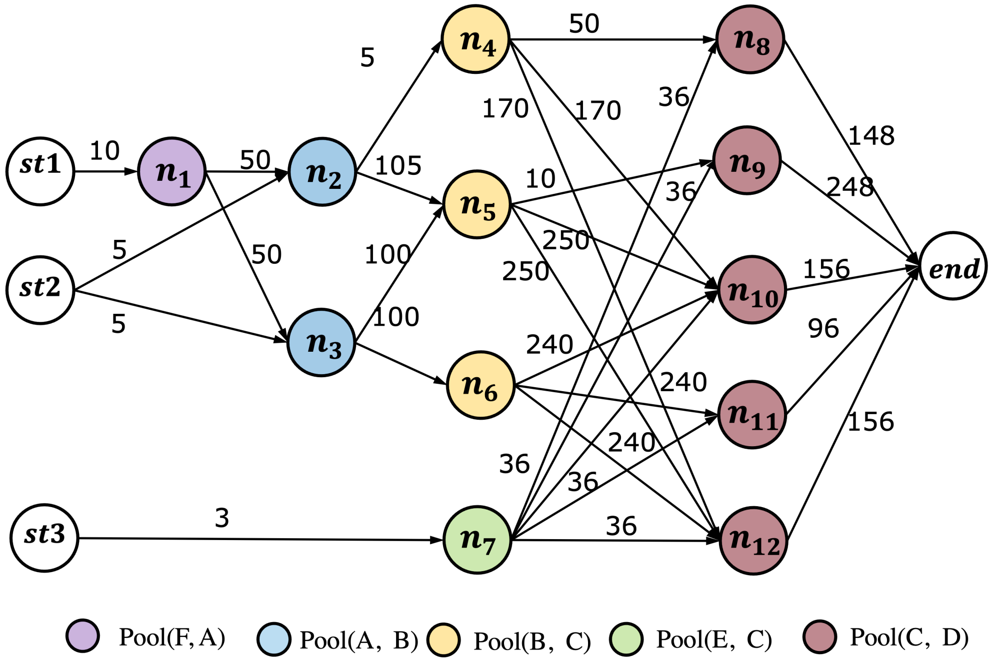
  $n_1.d_e < n_3.d_e$

  $prune\ n_i\ if:$

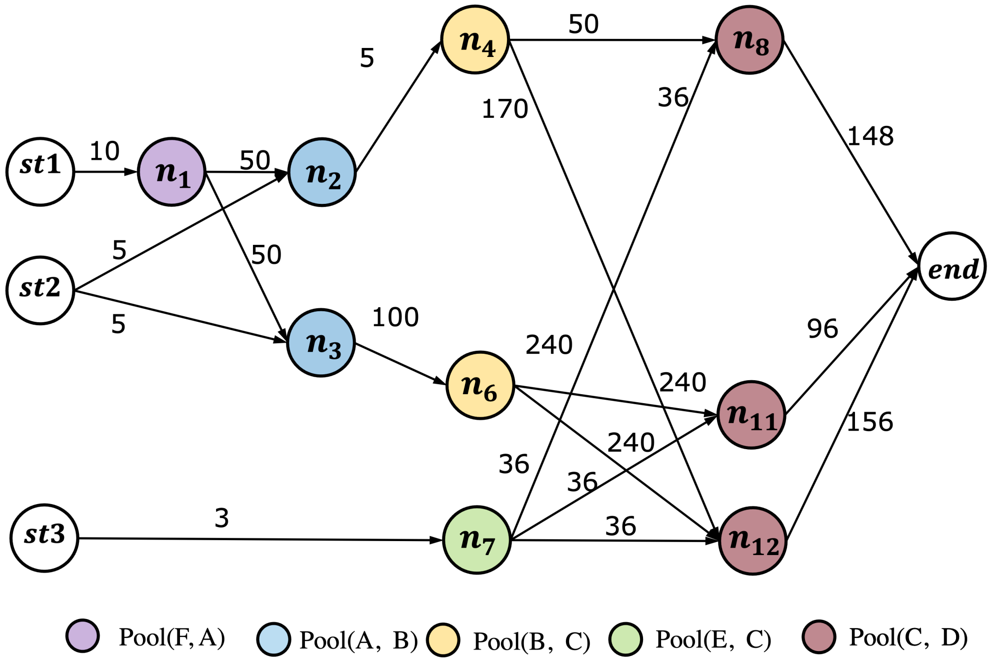
  $n_i\ has\ no\ outgoing\ edges$



Pruned Gloria Graph

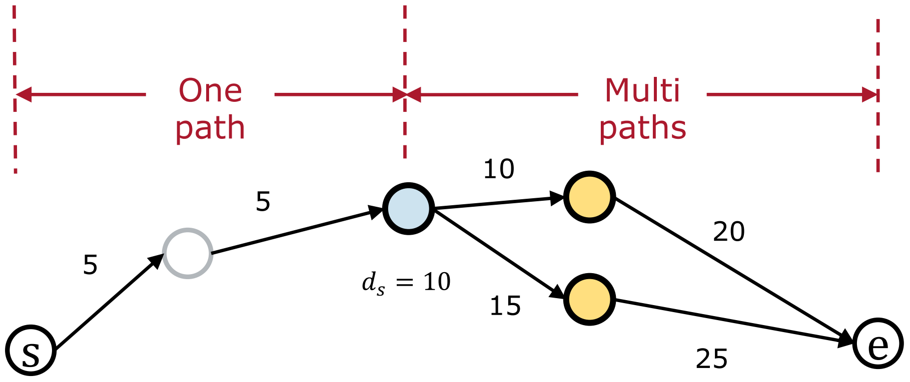


Full Gloria Graph



Pruned Gloria Graph

Optimal Path Search

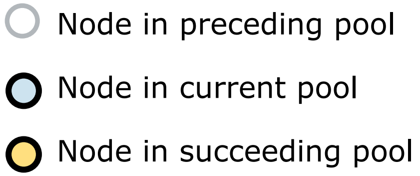


- Compute d_e

$$d_e = \min\{path_i.weight\} = 30$$

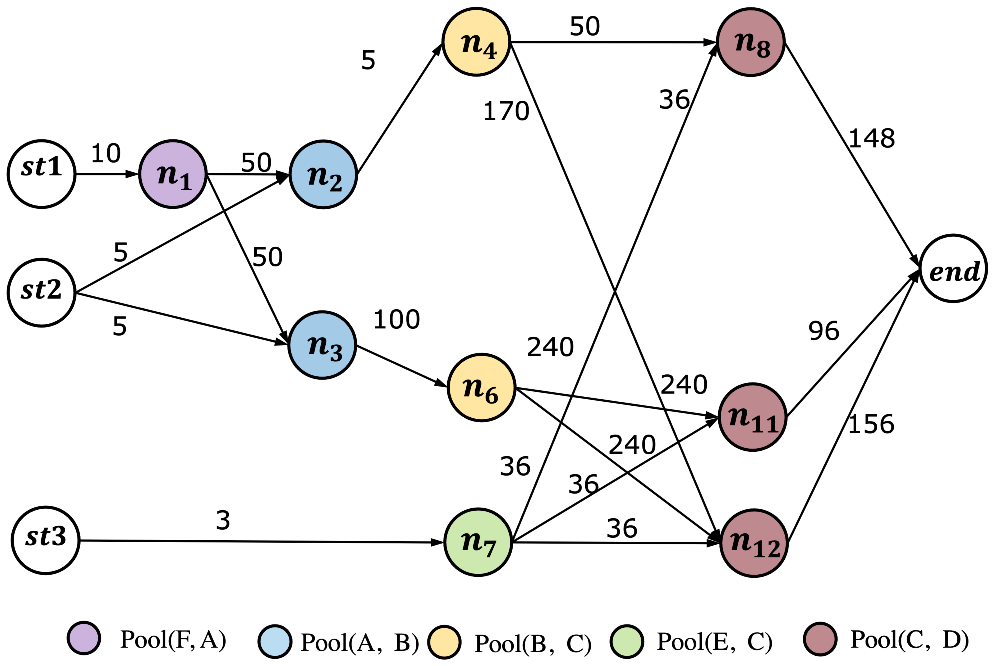
- Prune edges

- One node is passed by one complete path



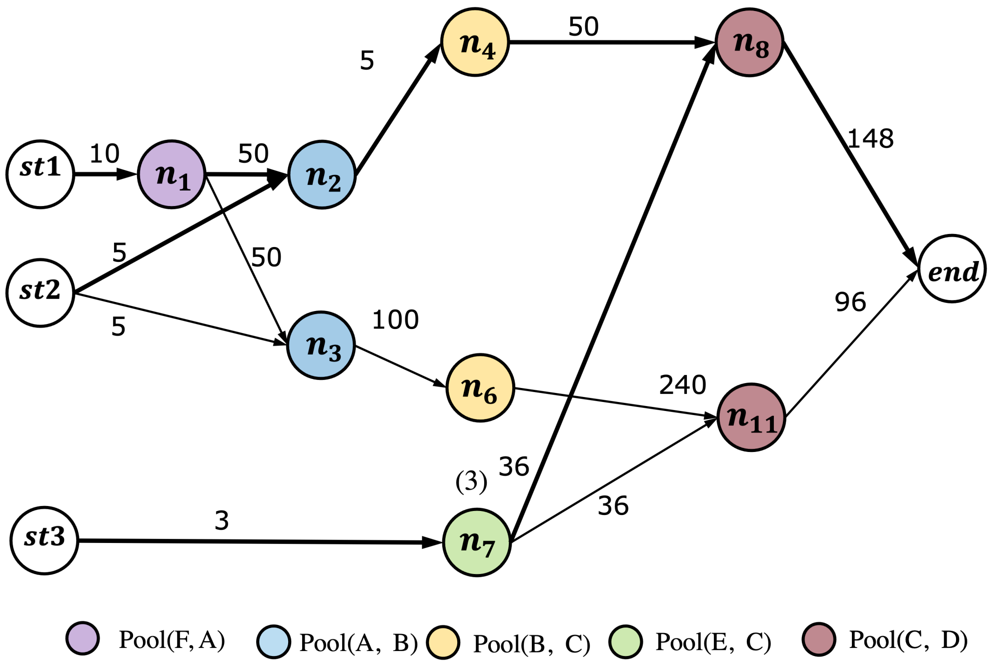
Path Search

4 paths



Pruned Gloria Graph

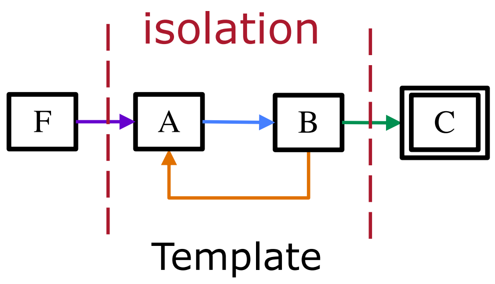
2 paths

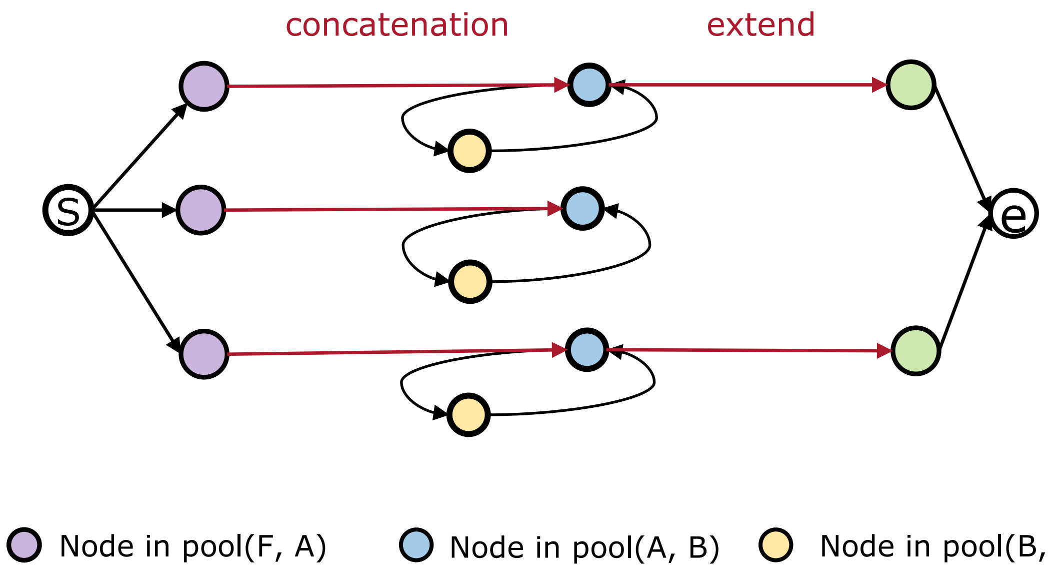


Reversely Pruned Gloria Graph

Gloria Optimizer For Kleene

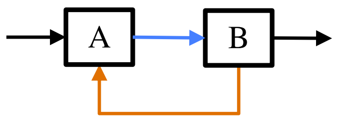
Kleene sub-graph





- Isolate
- Construct separately
- Concatenate
- Prune
- Extend

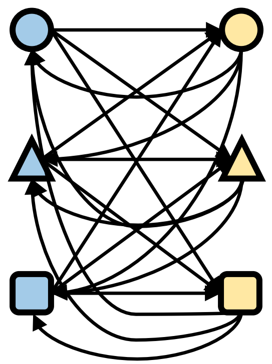
Cycle path



Template

- Edges from pool(A, B) to pool(B, A)
- Edges from pool(B, A) to pool(A, B)
- Consistent cycle paths

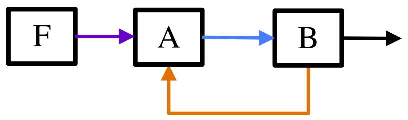
Lemma 6.1. For a flat or nested Kleene sub-pattern, in its Kleene sub-graph, a cycle path that has nodes with different sharing plans can be safely pruned.



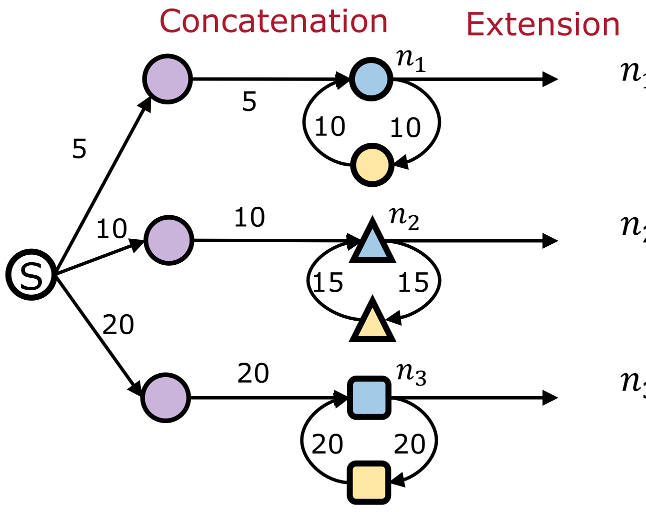
Node in pool(A, B) Node in pool(B, A)

△ ○ □ Different sharing plans

Pruning



Template



$n_1.d_s = 5 + 5 + 10 + 10 = 30$

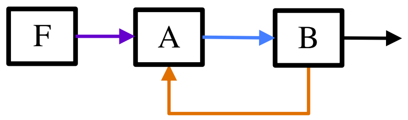
$n_2.d_s = 10 + 10 + 15 + 15 = 50$

$n_3.d_s = 20 + 20 + 20 + 20 = 80$

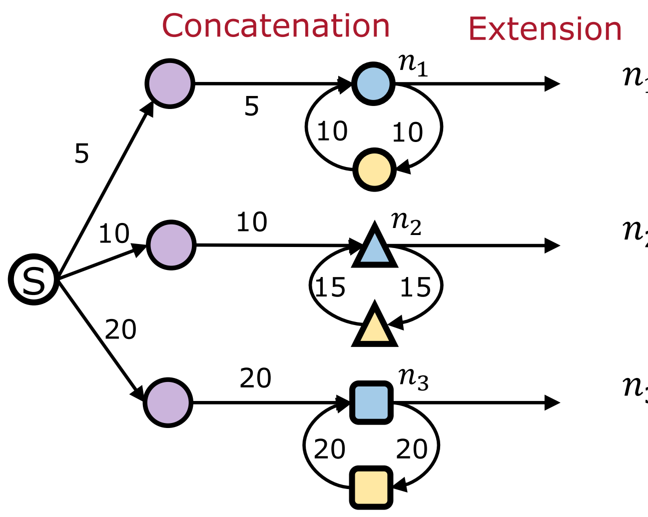
Node in pool(A, B) Node in pool(B, A)

△ ○ □ Different sharing plans

Pruning



Template



$n_1.d_s = 5 + 5 + 10 + 10 = 30$

$n_2.d_s = 10 + 10 + 15 + 15 = 50$

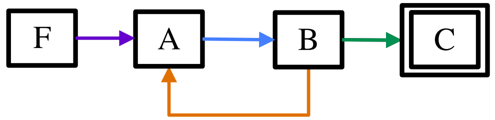
$n_3.d_s = 20 + 20 + 20 + 20 = 80$

Node in pool(A, B) Node in pool(B, A)

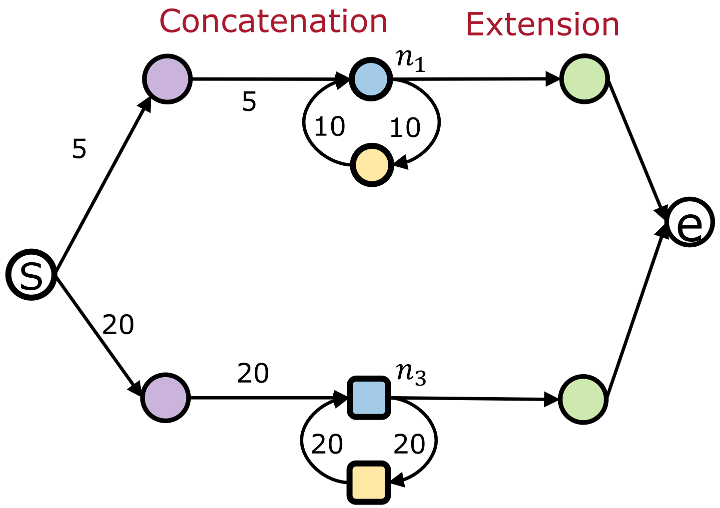
Different sharing plans

- Apply Node Pruning Rule

Graph Extension



Template



Node in pool(B, C) Node in pool(A, B) Node in pool(B, A)

Different sharing plans

Experimental Evaluation

Common Experiment Setting

Infrastructure

Java 8, Ubuntu 14.04, 16 cores, 128GB

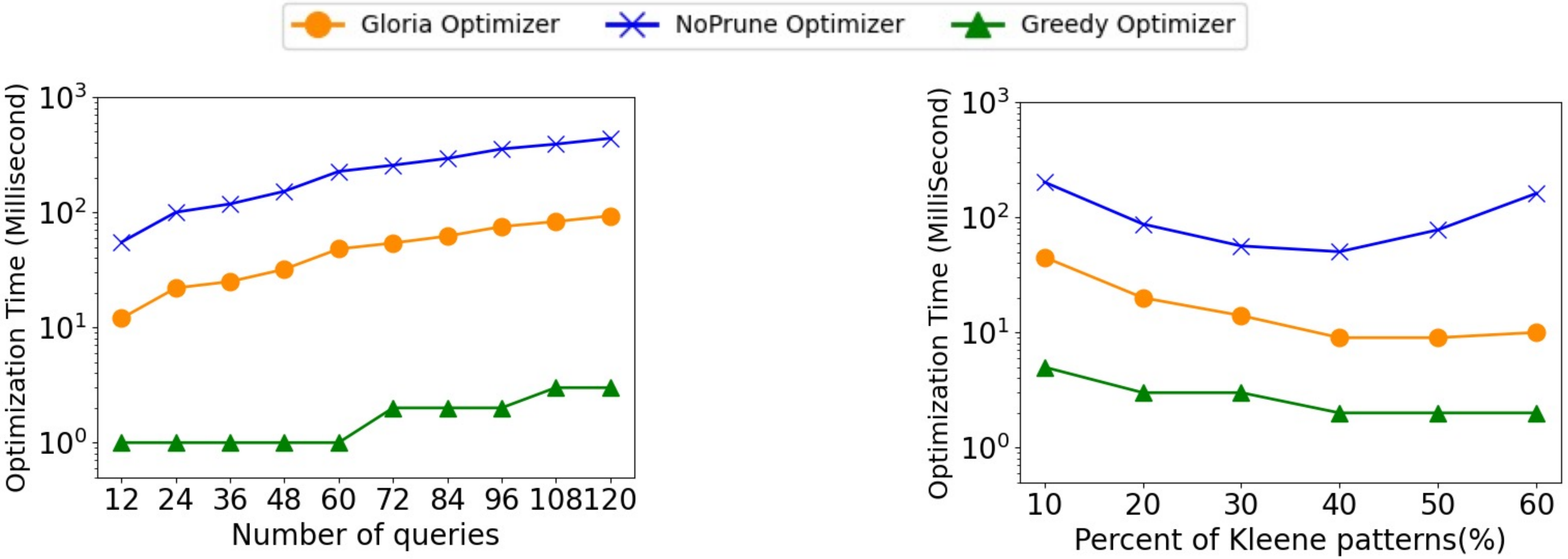
Data sets

- NYC taxi and Uber real data set
- Stock real data set
- Dublin bus data set

Experiment Sets

- Optimizer experiments
- Executor experiments

Optimization Efficiency

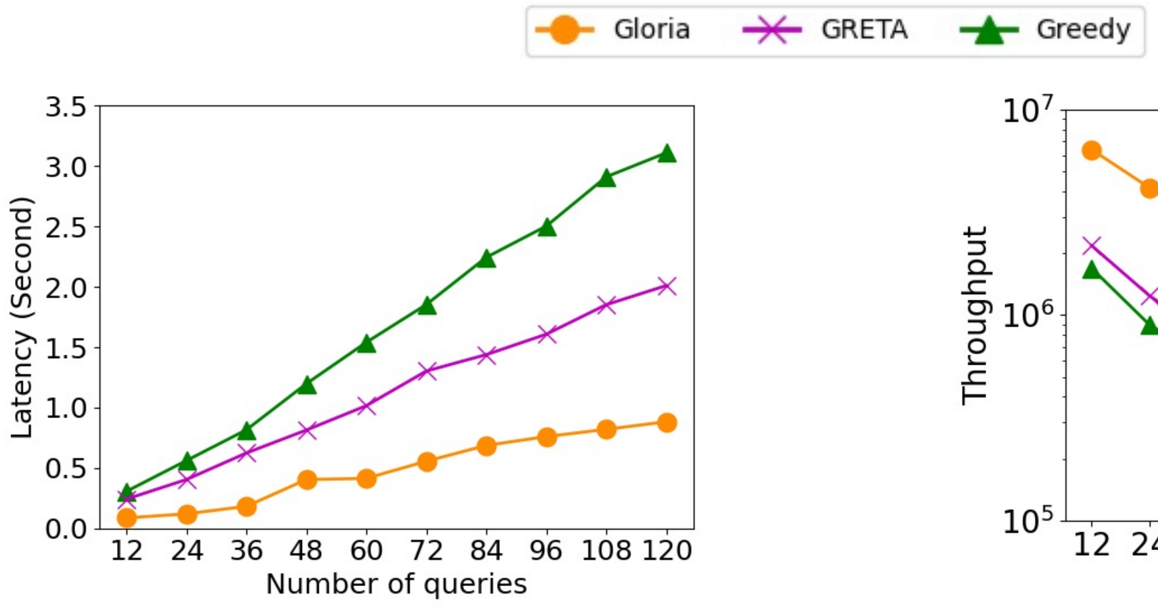


Bus data set (Mixed)

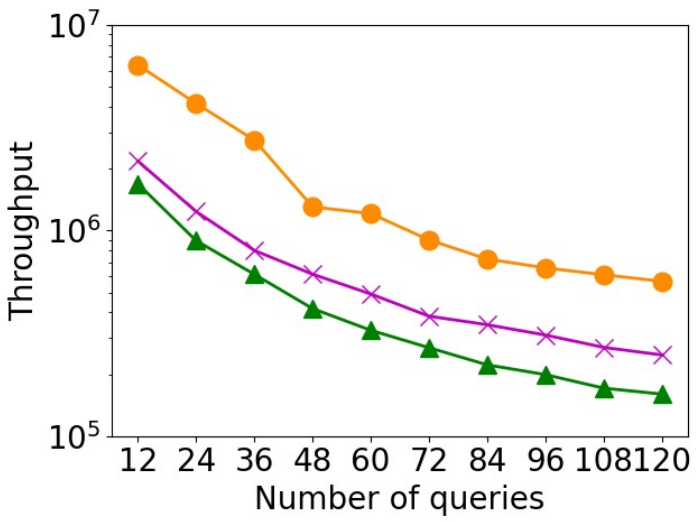
Bus data set (Mixed)

- Gloria Optimizer outperforms NoPrune Optimizer by a factor of 5 in the Mixed workload.
- Gloria Optimizer outperforms NoPrune Optimizer by 6-fold to 1.2 order of magnitude in the Mixed workload.

Optimization Quality



Latency (Stock)



Throughput (Stock)

- Gloria sharing plan outperforms Greedy sharing plan and GRETA by 10-fold and 3-fold respectively.

Conclusions

Gloria

- Introduces a Graph-based Sharing Optimizer for event trend aggregation.
- Constructs a Gloria Graph to transform the optimal workload sharing plan problem into an Optimal Path Search Problem.
- Proposes Pruning Rules to reduce the size of the gloria graph.
- Designs an efficient path search algorithm to Find the Optimal Path.
- Is demonstrated to achieve Significant Performance Gains over state-of-the-art.

Acknowledgement

Chuan Lei
Researcher

Olga Poppe
Researcher

Elke A. Rundensteiner
Professor

Funding agency:
- NSF grants IIS-1815866