Contigra: Graph Mining with Containment Constraints

Joanna Che

Kasra Jamshidi

Keval Vora

Simon Fraser University

EuroSys 2024

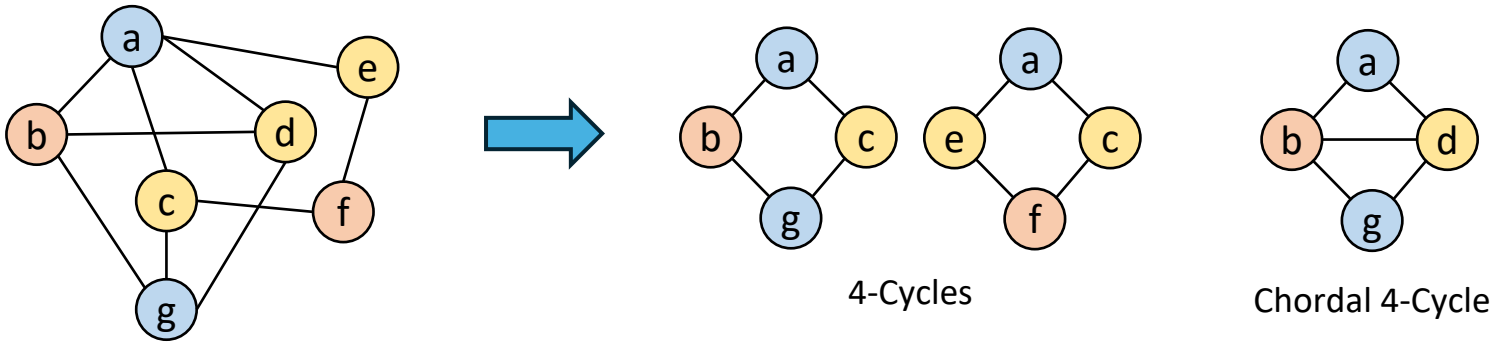
April 23, 2024

SFU

Contigra: Graph Mining with Containment Constraints

Graph Mining

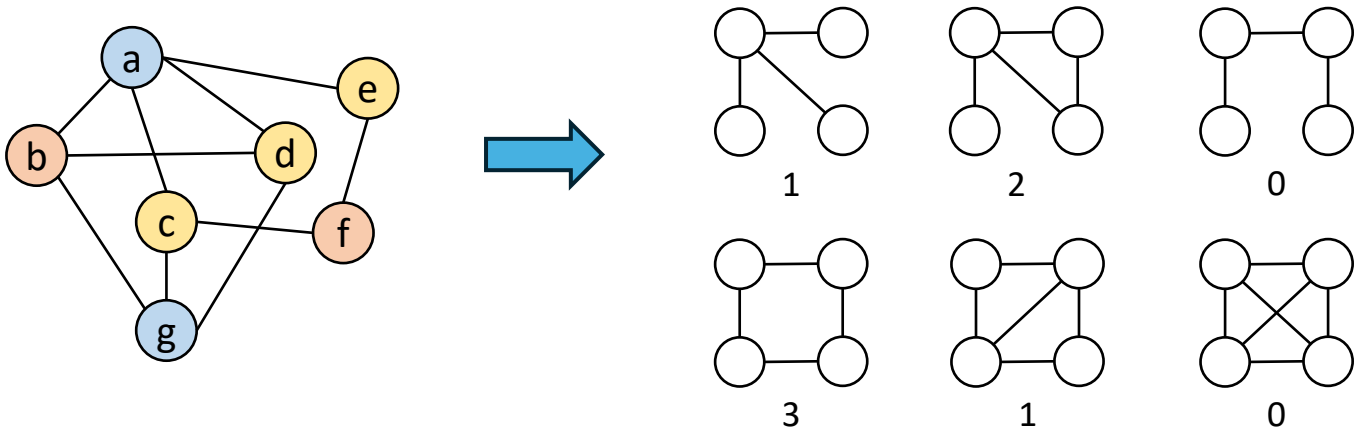
- Explore subgraphs of interest in large graphs
  - Subgraph isomorphism



SFU

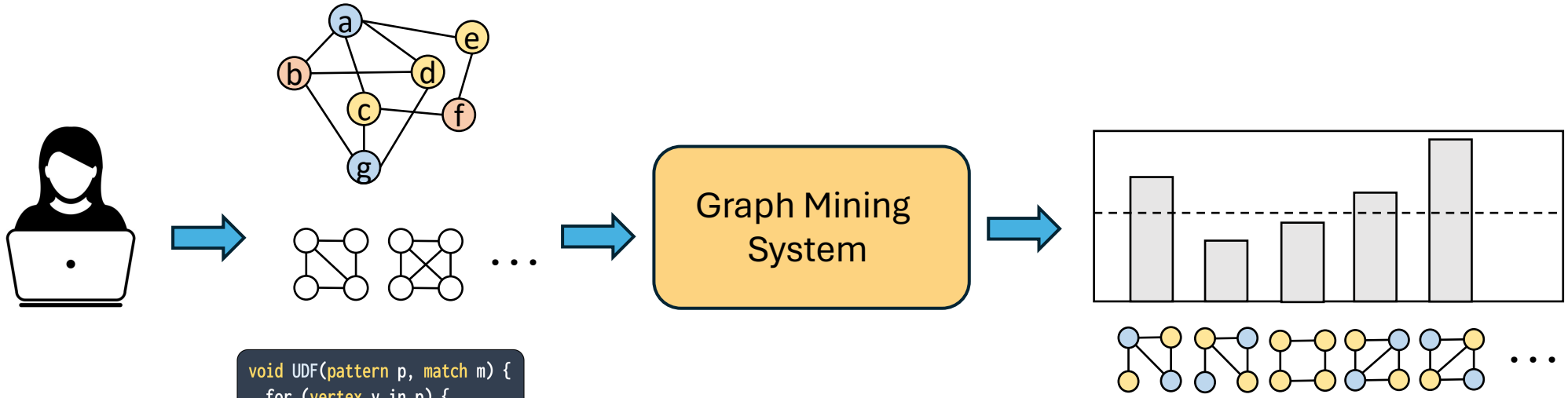
Graph Mining

- Explore subgraphs of interest in large graphs
  - Subgraph isomorphism



SFU

Graph Mining Systems

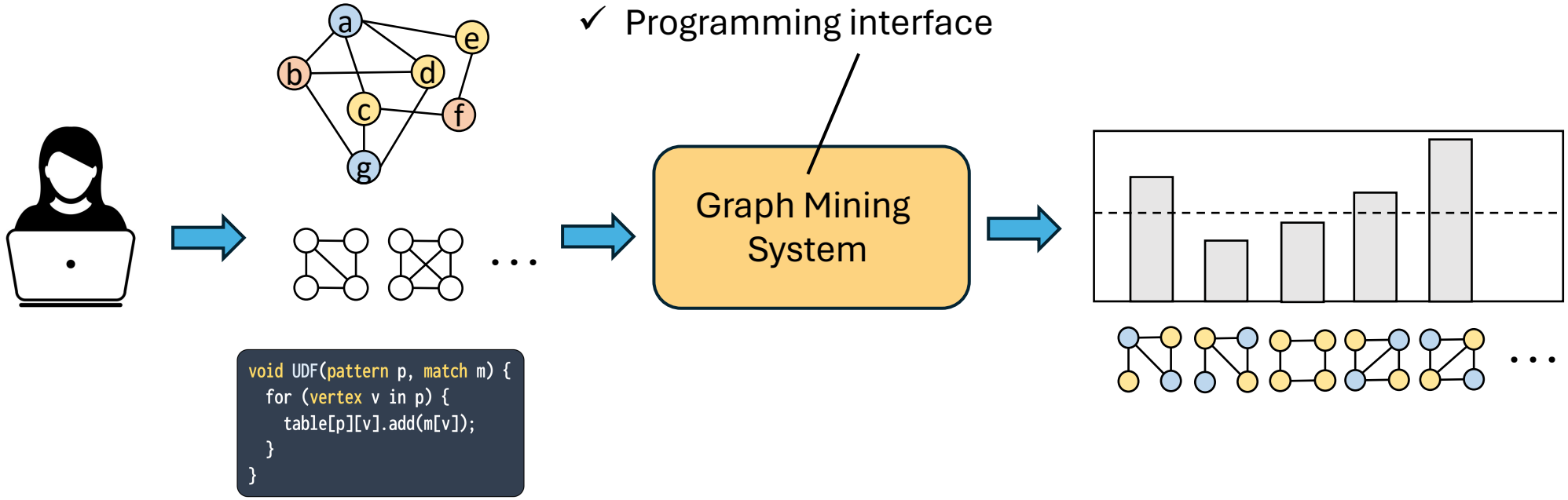



```
void UDF(pattern p, match m) {
  for (vertex v in p) {
    table[p][v].add(m[v]);
  }
}
```

SFU

Graph Mining Systems

- ✓ Pattern matching / graph theory
- ✓ Task-parallel & scalable
- ✓ Programming interface

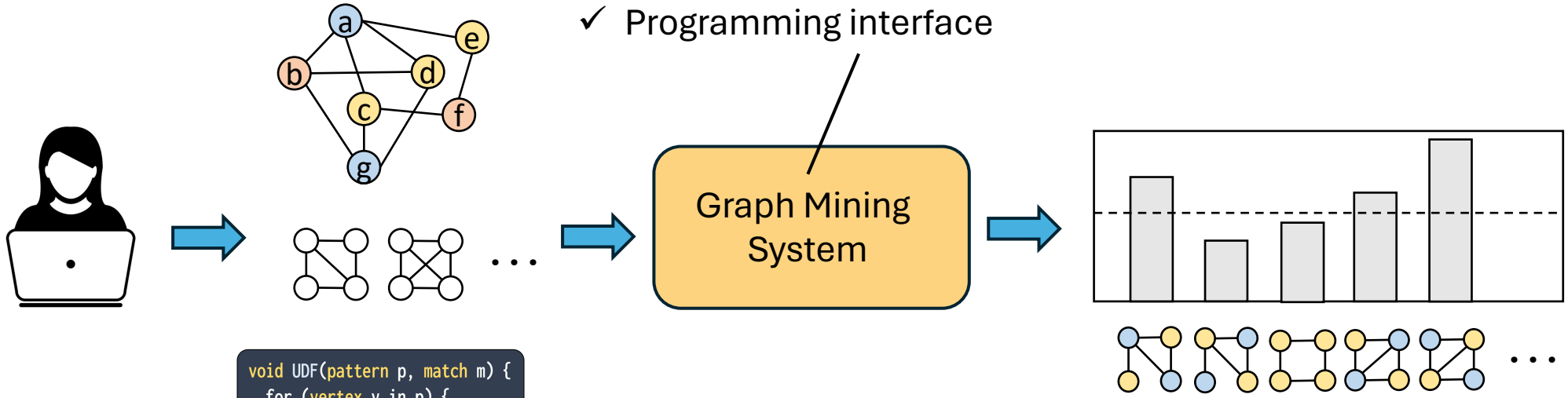



```
void UDF(pattern p, match m) {
  for (vertex v in p) {
    table[p][v].add(m[v]);
  }
}
```

SFU

Graph Mining Systems

- ✓ Pattern matching / graph theory
- ✓ Task-parallel & scalable
- ✓ Programming interface

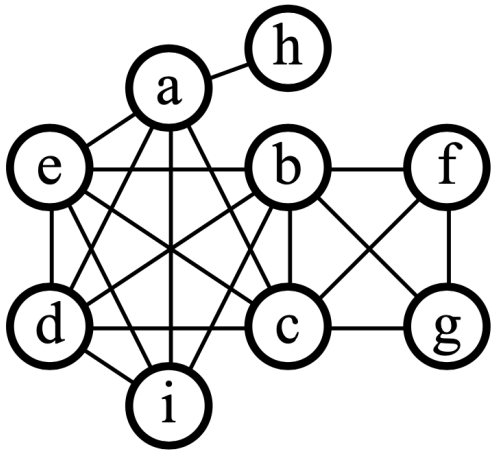


- Various applications: motif counting, frequent subgraph matching, pattern matching, etc.
- Additional constraints like maximality of subgraphs

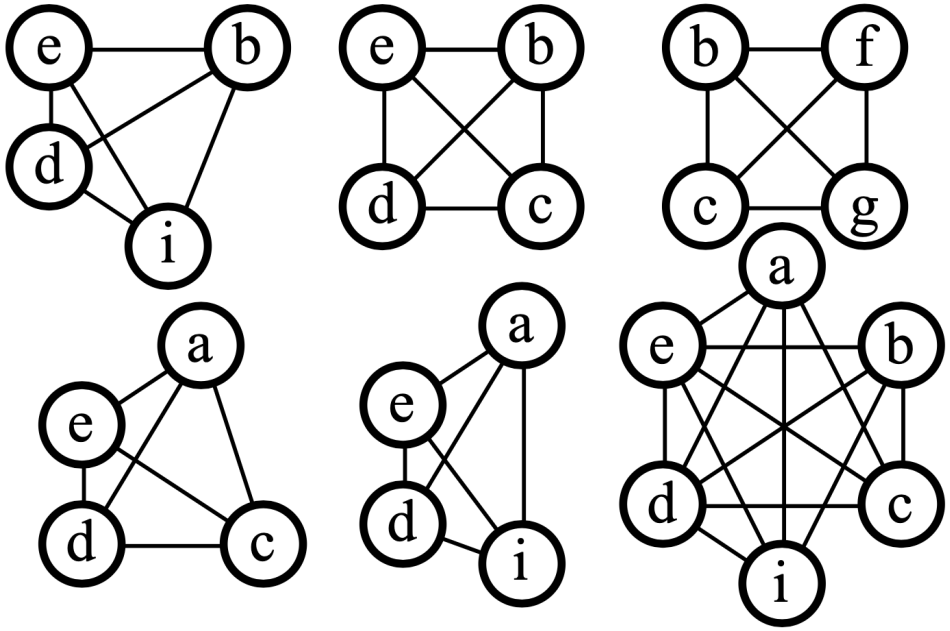
SFU

Graph Mining with Containment Constraints

- Containment constraint: whether a subgraph should be inside or outside another subgraph
- **Maximality**: subgraph should not be inside another



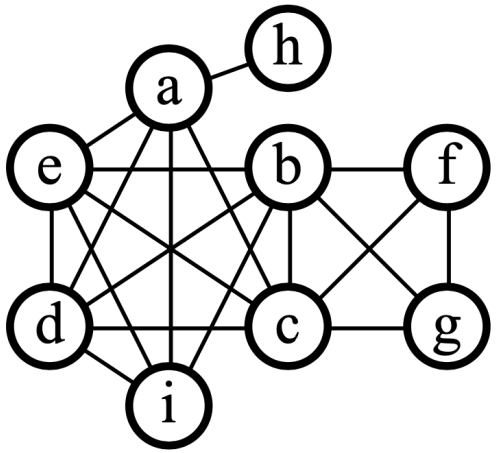
Maximal Quasi-Cliques



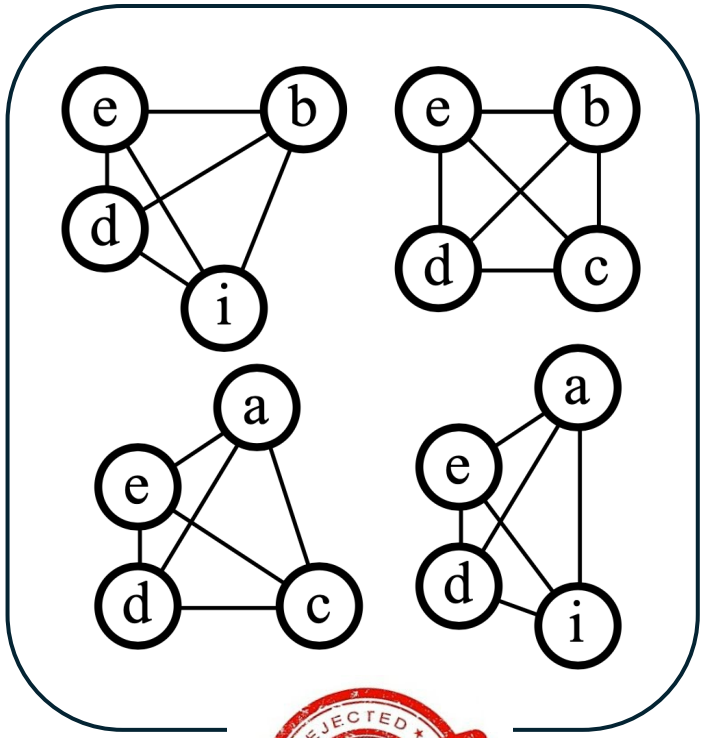
SFU

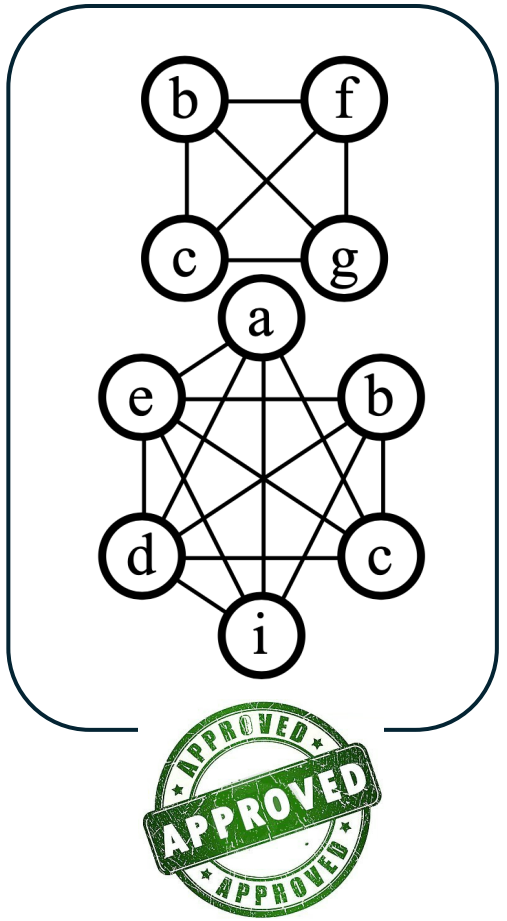
Graph Mining with Containment Constraints

- Containment constraint: whether a subgraph should be inside or outside another subgraph
- **Maximality**: subgraph should not be inside another



Maximal Quasi-Cliques

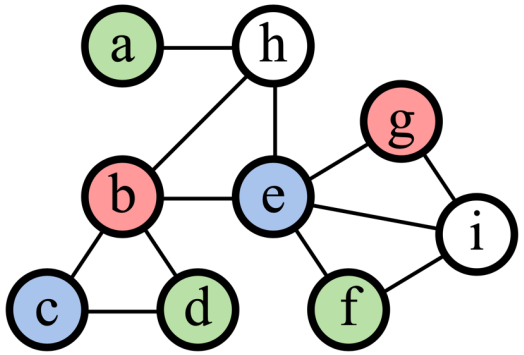




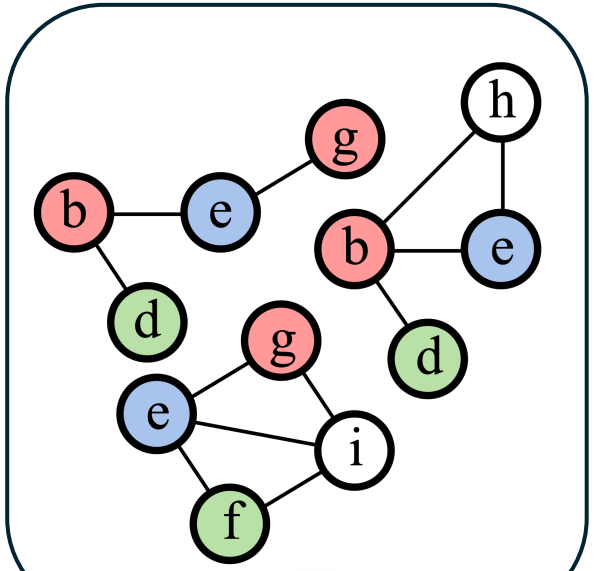
SFU

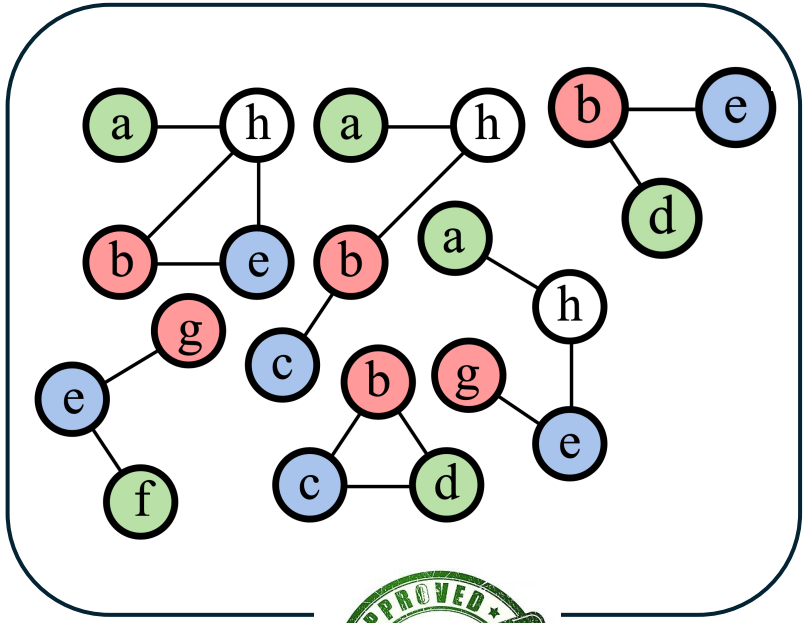
Graph Mining with Containment Constraints

- Containment constraint: whether a subgraph should be inside or outside another subgraph
- Maximality: subgraph should not be inside another
- **Minimality**: subgraph should not contain another



Minimal Subgraphs covering: {●●●}

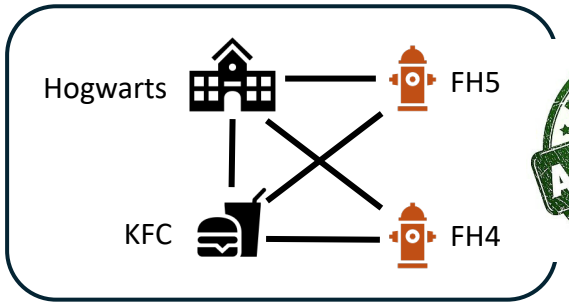




SFU

Graph Mining with Containment Constraints

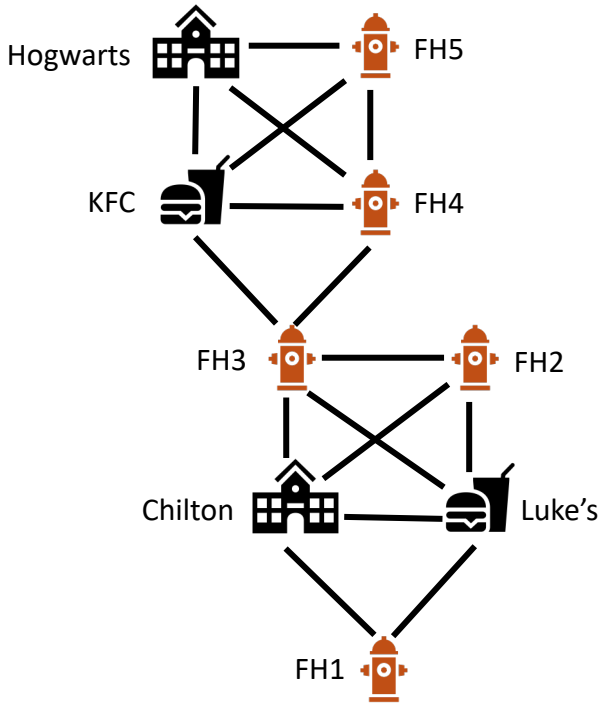
- Containment constraint: whether a subgraph should be inside or outside another subgraph
- Maximality: subgraph should not be inside another
- Minimality: subgraph should not contain another
- **Nested Subgraph Queries, Anti-Vertex Queries [*]**



Find all  with ***exactly*** 2 

```
MATCH (a:SCHOOL)--(b:BUSINESS),
      (a)--(c:FIRE_HYDRANT)--(b),
      (a)--(d:FIRE_HYDRANT)--(b)
WHERE NOT EXISTS {
  MATCH (a:SCHOOL)--(b:BUSINESS),
        (a)--(c:FIRE_HYDRANT)--(b),
        (a)--(d:FIRE_HYDRANT)--(b),
        (a)--(e:FIRE_HYDRANT)--(b)
}
RETURN a, b, c, d
```

Cypher Query

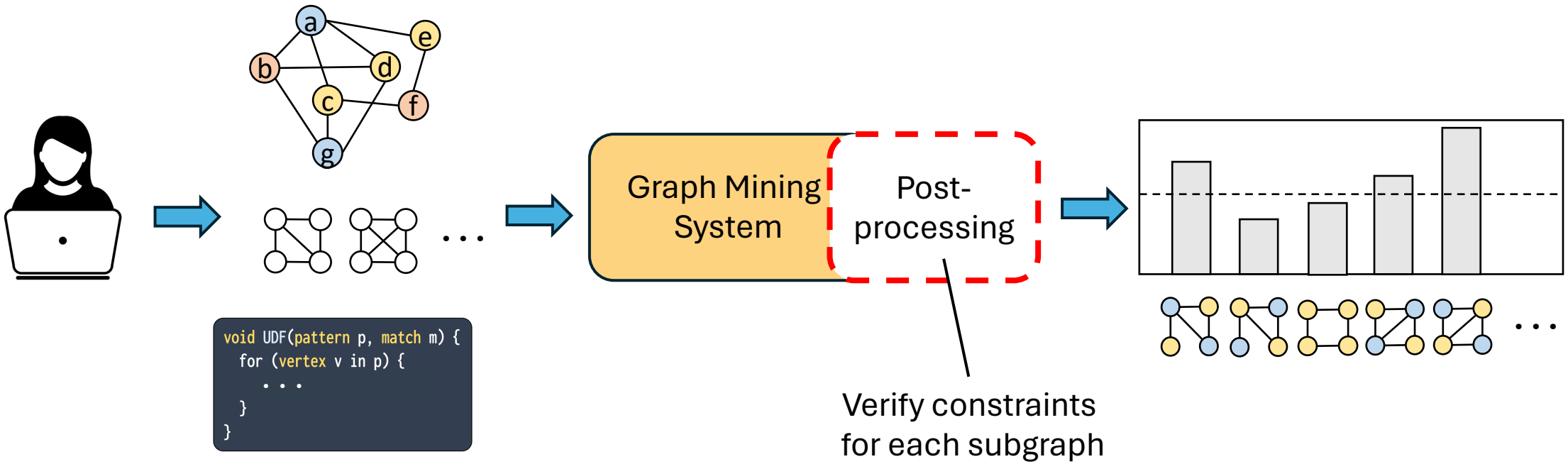


[*] Anti-Vertex for Neighborhood Constraints in Subgraph Queries. SIGMOD GRADES-NDA 2022.

SFU

Challenge with Containment Constraints

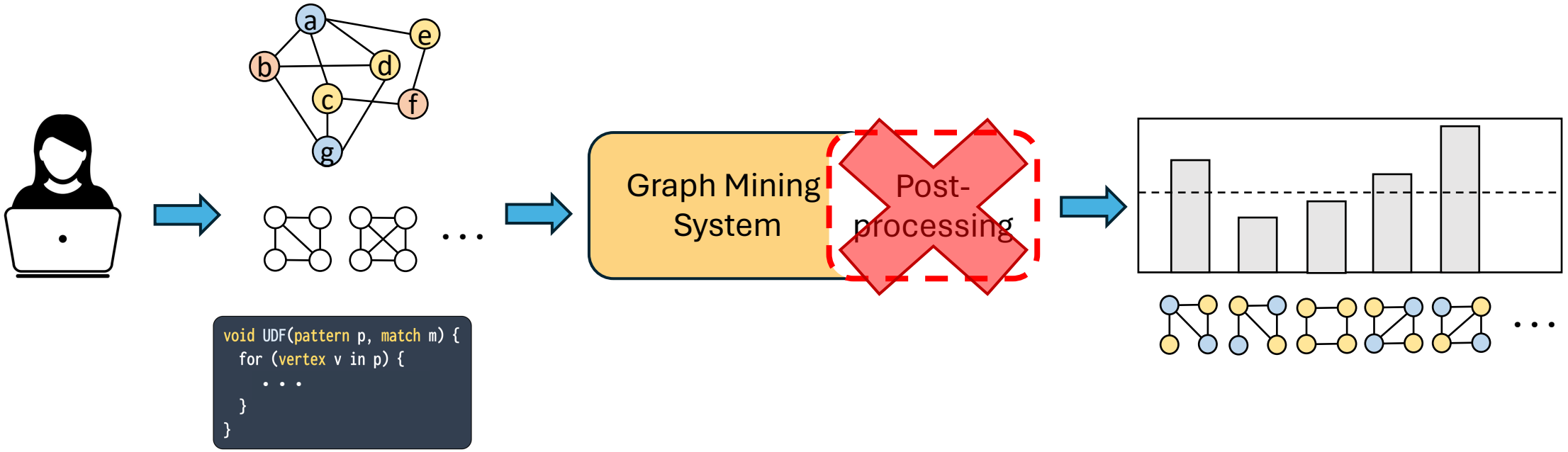
- **Unnecessary** subgraph computation
- **Expensive checking after** each subgraph is explored



SFU

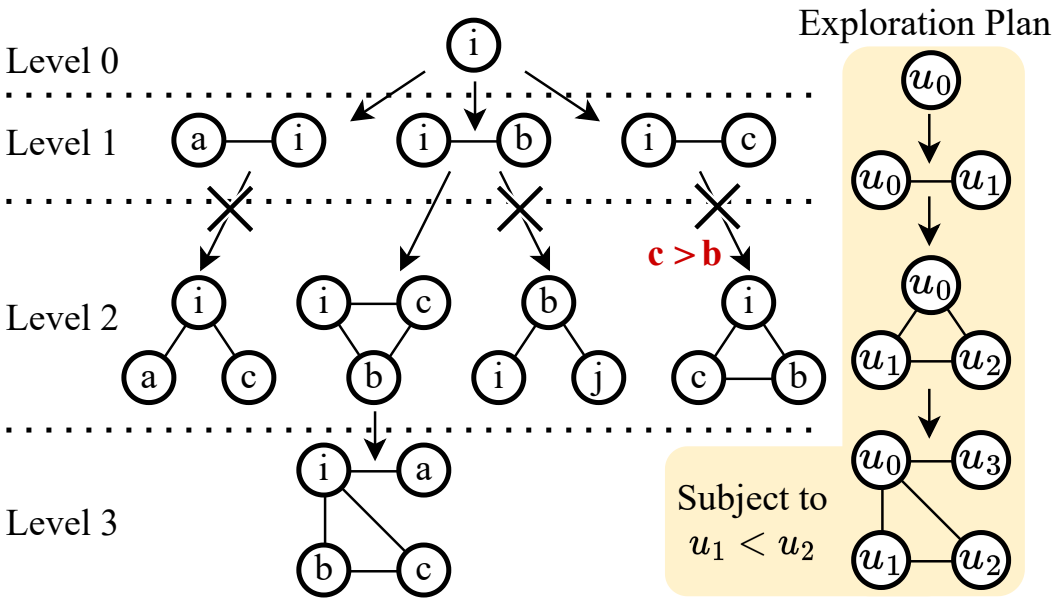
Contigra for Containment Constraints

- Directly explore subgraphs that satisfy required constraints ✓ Eliminate unnecessary computation
- Execution model aware of containment constraints ✓ Simplify constraint checking



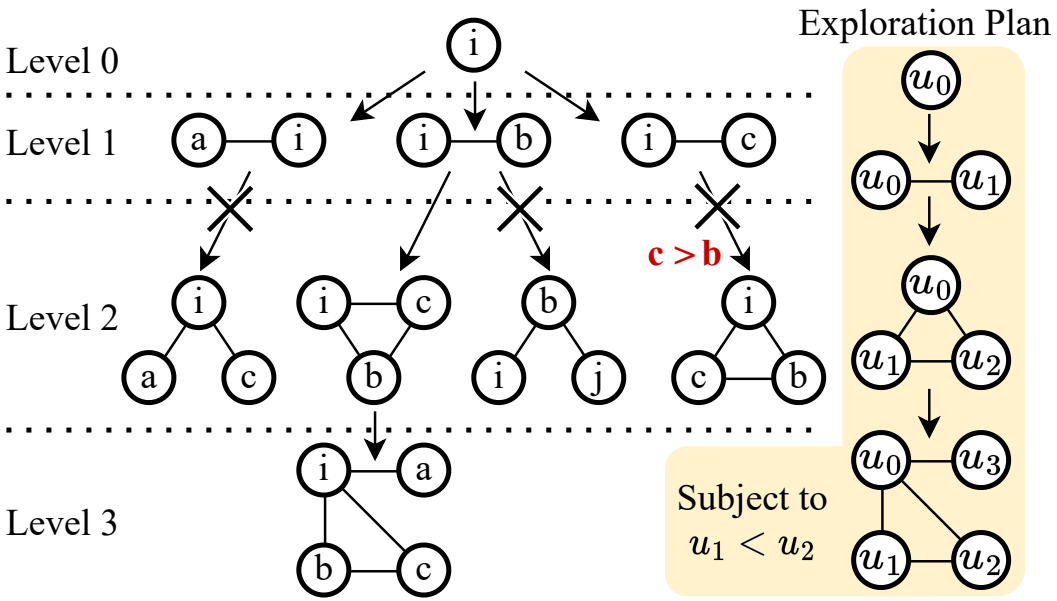
Subgraph Exploration Tasks

- Static, independent exploration tasks that traverse the graph in parallel



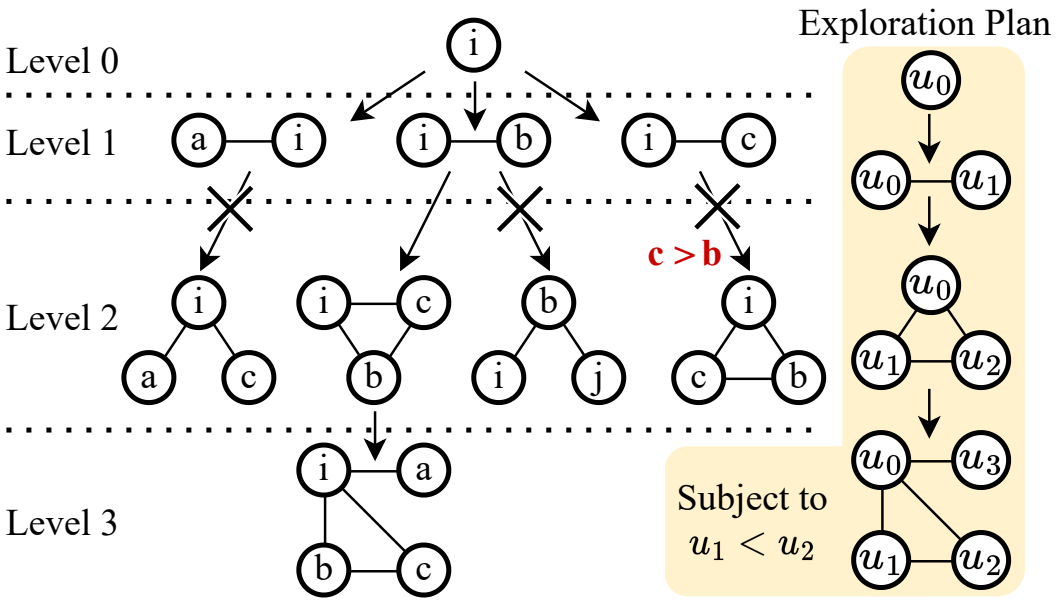
Subgraph Exploration Tasks

- Static, independent exploration tasks that traverse the graph in parallel
- Depth-first exploration following pattern-specific exploration plan ➡ **Search Tree**



Subgraph Exploration Tasks

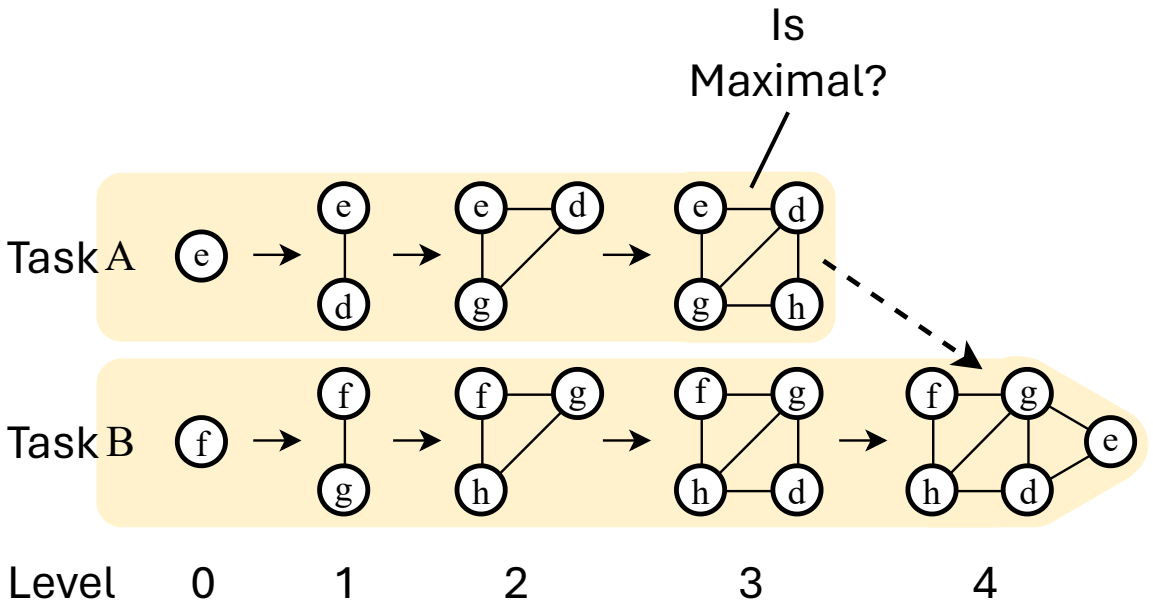
- Static, independent exploration tasks that traverse the graph in parallel
- Depth-first exploration following pattern-specific exploration plan ➡ **Search Tree**
- **Task-local Cache** maintains potential mappings for vertices at each level



Cross-Task Dependencies from Containment Constraints

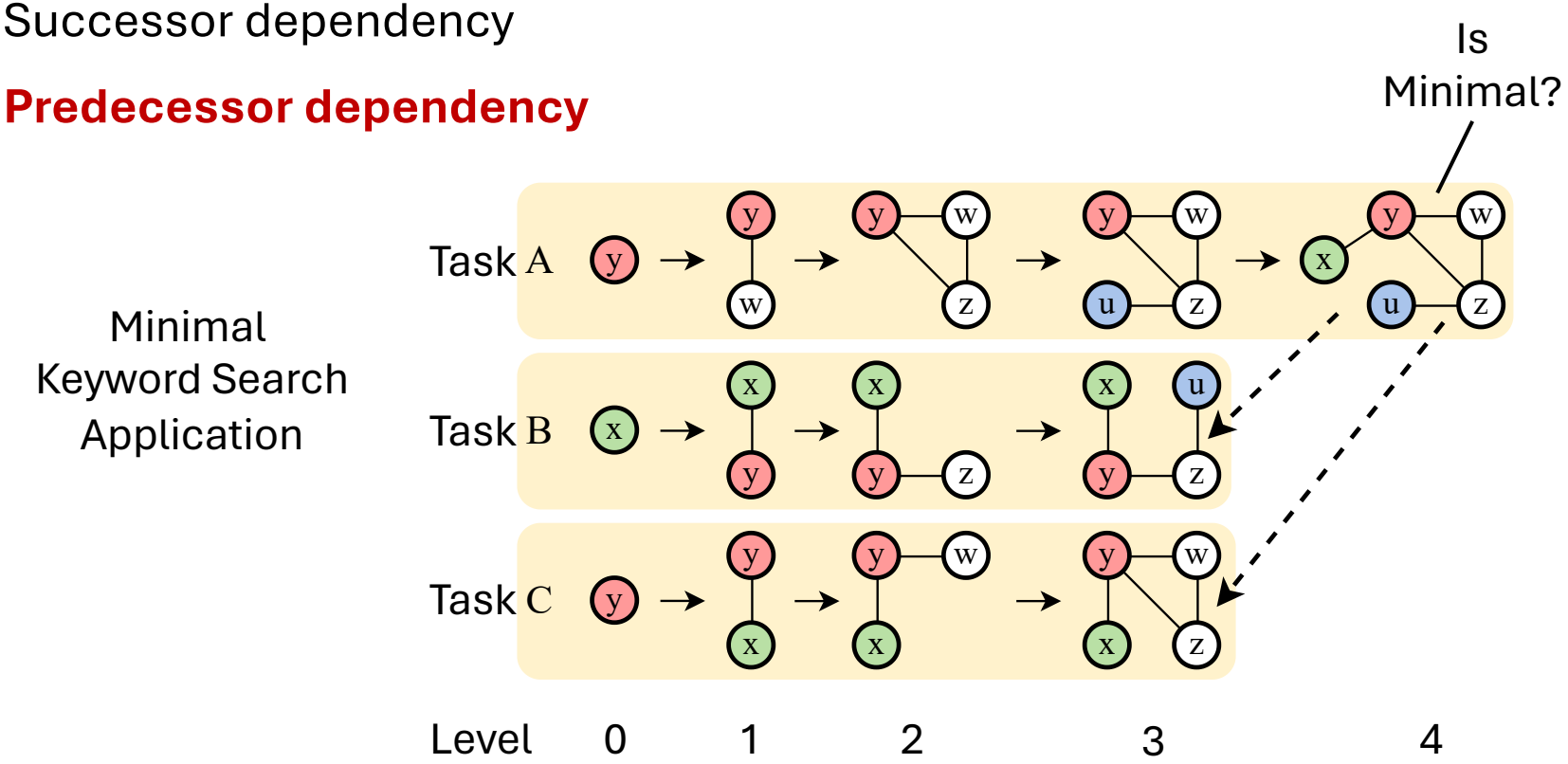
- **Successor dependency**

Maximal Quasi-Cliques Application



Cross-Task Dependencies from Containment Constraints

- Successor dependency
- **Predecessor dependency**



Cross-Task Dependencies from Containment Constraints

- Successor dependency
- Predecessor dependency
- **Lateral dependency**
  - Automatically inferred across specific tasks

Contigra Execution Model

- Validation Tasks: special constraint-checking tasks

Contigra Execution Model

- Validation Tasks: special constraint-checking tasks
- Task management strategies leverage available dependencies

- Task Fusion
- Task Promotion

Successor Dependencies

- Task Cancelation

Lateral Dependencies

- Task Skipping
- Eager Filtering

Predecessor Dependencies

Contigra Execution Model

- **Validation Tasks**: special constraint-checking tasks
- Task management strategies leverage available dependencies

- **Task Fusion**
- **Task Promotion**

Successor Dependencies

- Task Cancelation

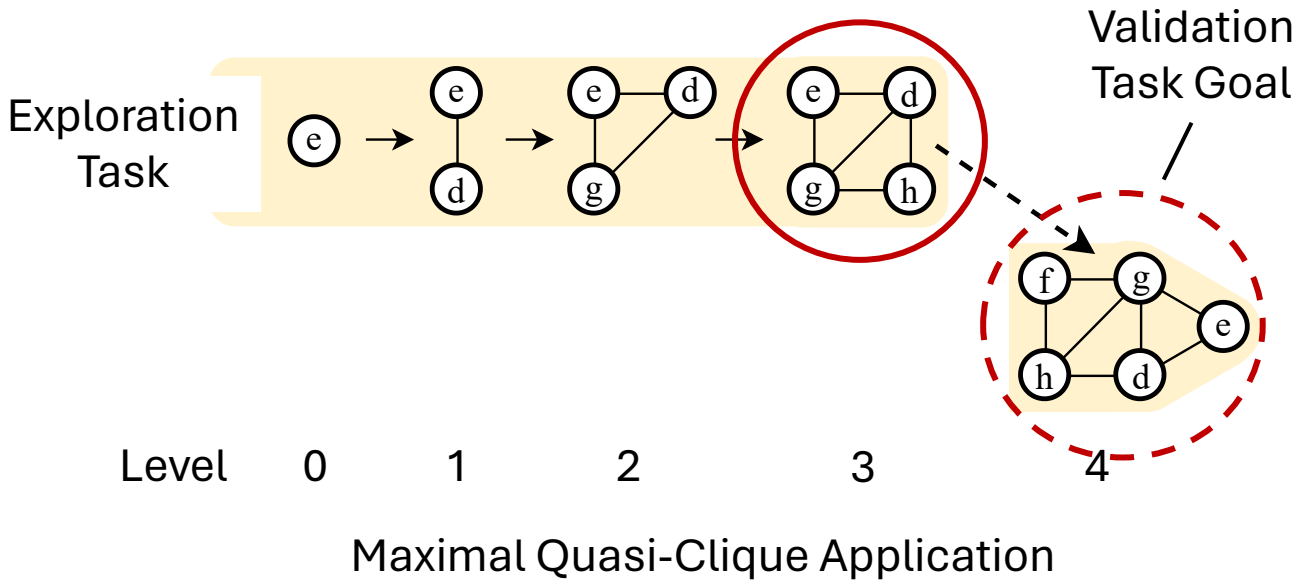
Lateral Dependencies

- Task Skipping
- Eager Filtering

Predecessor Dependencies

Validation Tasks

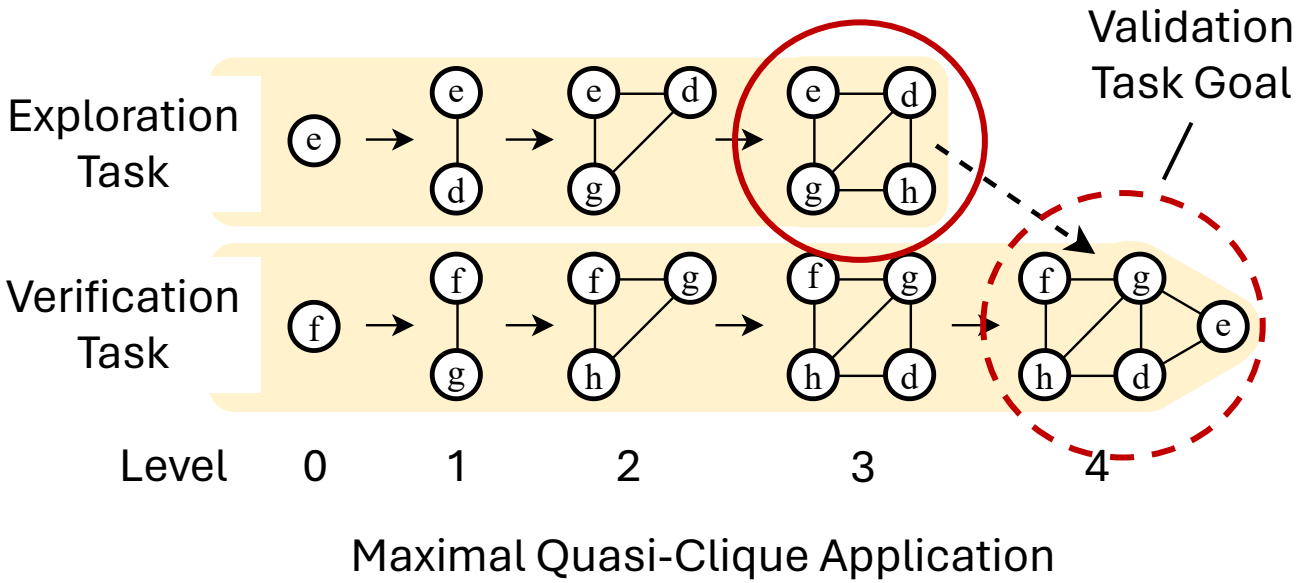
- Special tasks to validate subgraphs **on-the-fly** by exploring the constraining subgraphs



Validation Tasks

- Special tasks to validate subgraphs **on-the-fly** by exploring the constraining subgraphs
- Similar to exploration tasks, except:
  - Search subgraphs that contain specific vertices and edges
  - Terminate as soon as a subgraph is found

✓ Eliminate unnecessary computation

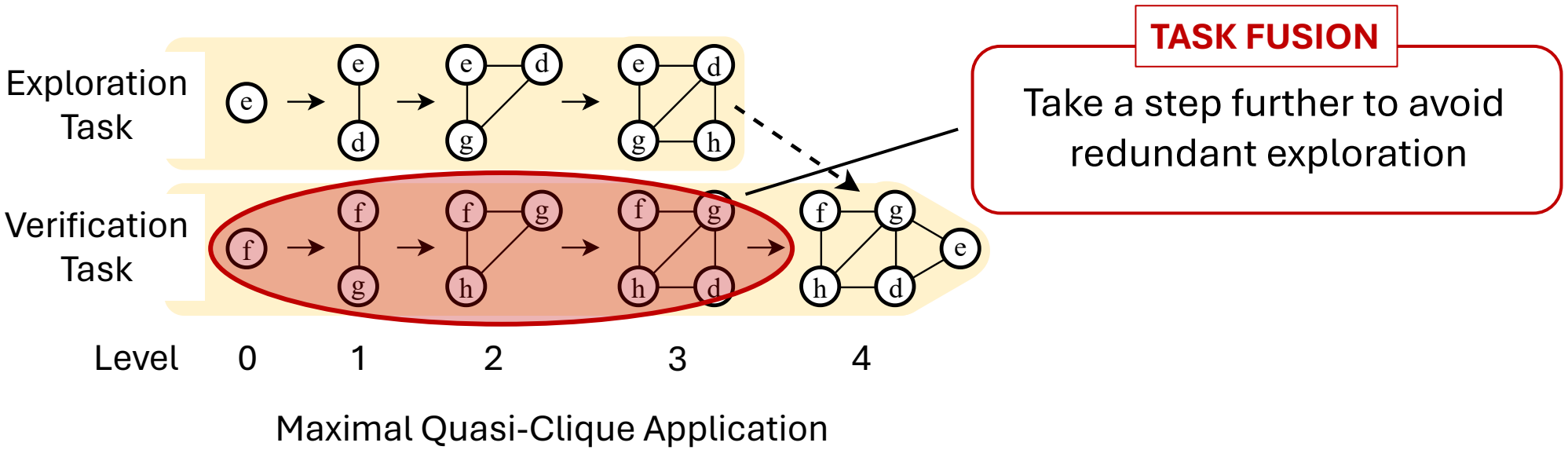


Maximal Quasi-Clique Application

Validation Tasks

- Special tasks to validate subgraphs **on-the-fly** by exploring the constraining subgraphs
- Similar to exploration tasks, except:
  - Search subgraphs that contain specific vertices and edges
  - Terminate as soon as a subgraph is found

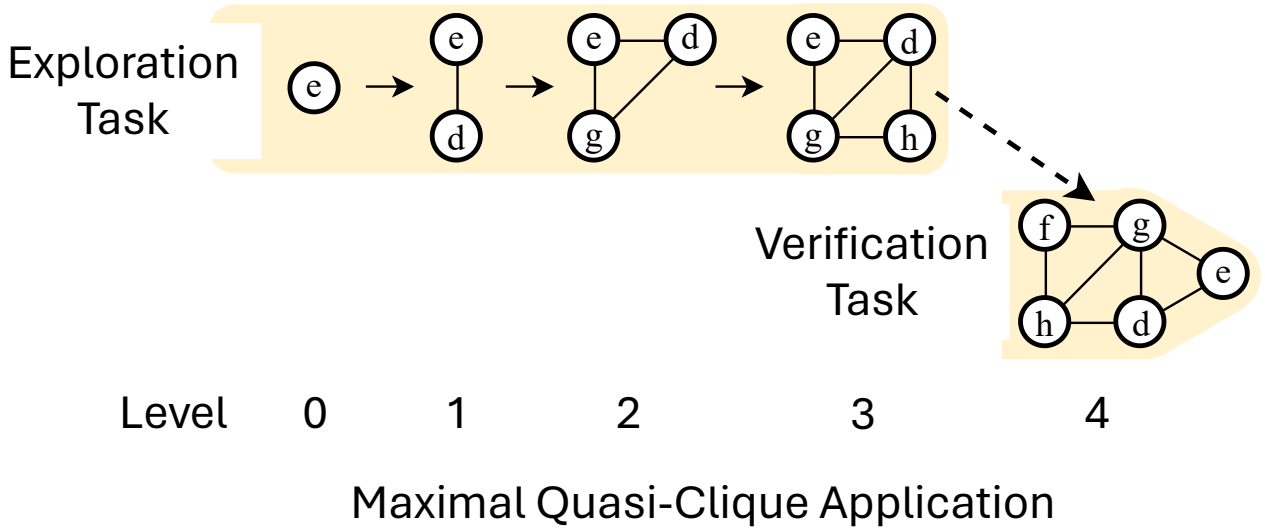
✓ Eliminate unnecessary computation



SFU

Task Fusion

- Fuse validation tasks with the exploration task that spawned them
- Eliminate redundant computations since the starting state consists:
  - Subgraph from exploration task
  - Task-local caches that maintain potential mappings for vertices at each level



TASK ALIGNMENT

Handle incompatible exploration plans

BRIDGING GAPS

Explore beyond consecutive levels in search tree

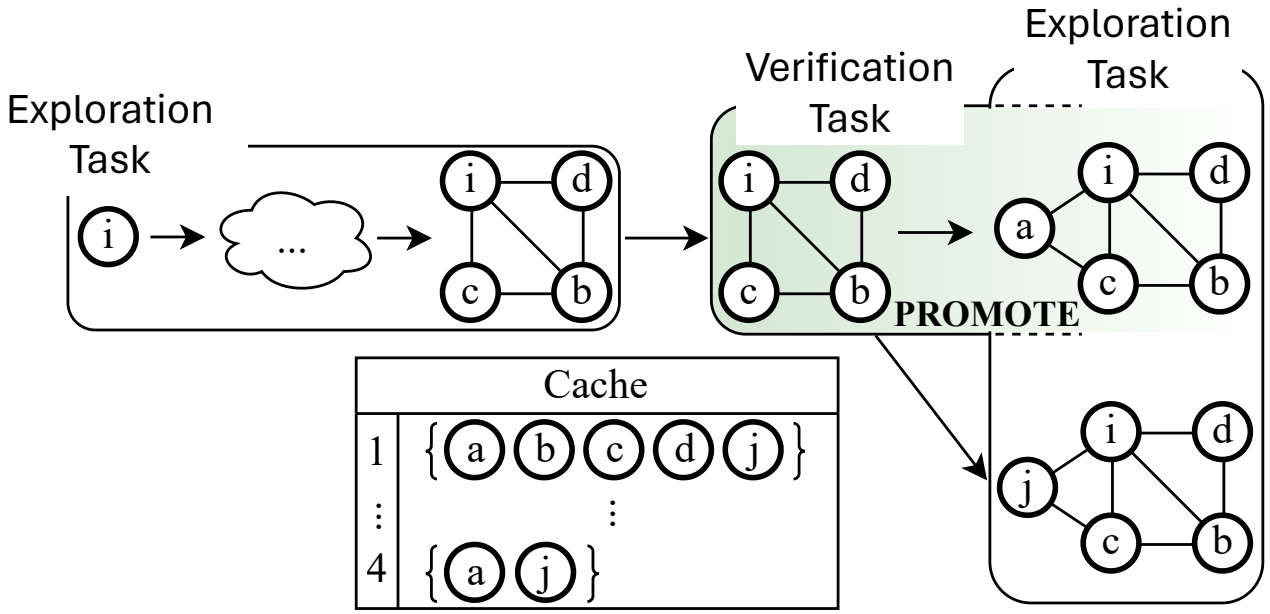
Validation Tasks & Task Fusion

- **Eliminates unnecessary computations** by verifying constraints on-the-fly
- **Reduces redundant explorations** by reusing subgraphs and Task-local Caches

Can we allow Validation Tasks
to continue as Exploration Tasks?

Task Promotion

- Promote validation tasks to subsequent exploration tasks
- Reuse the subgraph and Task-local Caches



Contigra Execution Model

- **Validation Tasks**: special constraint-checking tasks
- Task management strategies leverage available dependencies

- **Task Fusion**
- **Task Promotion**

Successor Dependencies

- Task Cancelation

Lateral Dependencies

- Task Skipping
- Eager Filtering

Predecessor Dependencies

SFU

Contigra Execution Model

- **Validation Tasks**: special constraint-checking tasks
- Task management strategies leverage available dependencies

- Task Fusion
- Task Promotion

Successor Dependencies

- **Task Cancelation**

Lateral Dependencies

- Task Skipping
- Eager Filtering

Predecessor Dependencies

Avoid unnecessary Validation Tasks via serial ordering

Contigra Execution Model

- **Validation Tasks**: special constraint-checking tasks
- Task management strategies leverage available dependencies

- Task Fusion
- Task Promotion

Successor Dependencies

- Task Cancelation

Lateral Dependencies

- **Task Skipping**
- **Eager Filtering**

Predecessor Dependencies

Avoid unnecessary exploration tasks by analyzing potential state spaces

Contigra Execution Model

- Validation Tasks: special constraint-checking tasks
- Task management strategies leverage available dependencies

- Task Fusion
- Task Promotion

Successor Dependencies

- Task Cancelation

Lateral Dependencies

- Task Skipping
- Eager Filtering

Predecessor Dependencies

Experimental Setup

- Implemented Contrigra on Peregrine+ (state-of-the-art multi-pattern exploration)
- Applications: Maximal Quasi-Cliques, Minimal Keyword Search, Nested Subgraph Queries, Quasi-Cliques w/o maximality

Data Graphs	Vertices	Edges	Labels
Amazon (AZ)	334.9K	925.9K	0
DBLP (DB)	317.1K	1.0M	0
Mico (MI)	96.6K	1.1M	28
Patents (PA)	2.7M	14.0M	36
Youtube (YT)	7.7M	50.7M	23
Products (PR)	2.4M	61.9M	46

- Baselines
  - Peregrine+ [EuroSys 2020] – general-purpose graph mining
  - TThinker [VLDB 2021] – specialized maximal quasi-cliques application

Performance Summary

Application	Baseline	Speedup
Maximal Quasi-Cliques	TThinker	12-41700x
Nested Subgraph Queries	Peregrine+	5.6-379x
Keyword Search	Peregrine+	21-16000x
Quasi-Cliques	Peregrine+	2.4-7.2x

Performance Summary

Application	Baseline	Speedup	Baseline Failures
Maximal Quasi-Cliques	TThinker	12-41700x	47%
Nested Subgraph Queries	Peregrine+	5.6-379x	7%
Keyword Search	Peregrine+	21-16000x	12%
Quasi-Cliques	Peregrine+	2.4-7.2x	-

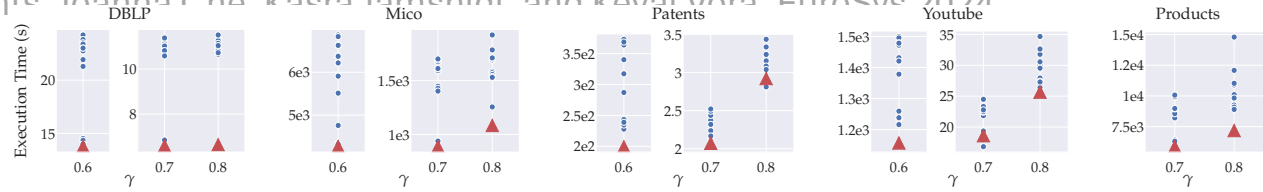
Task Fusion & Promotion

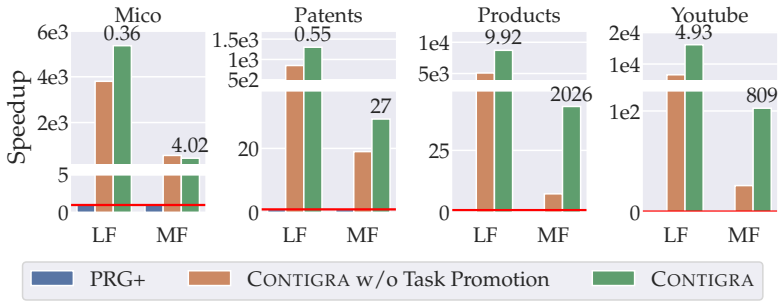
- Increases Task-local Cache utilization to 75%
- Eliminates 72% of Exploration Tasks & 77% of Validation Tasks

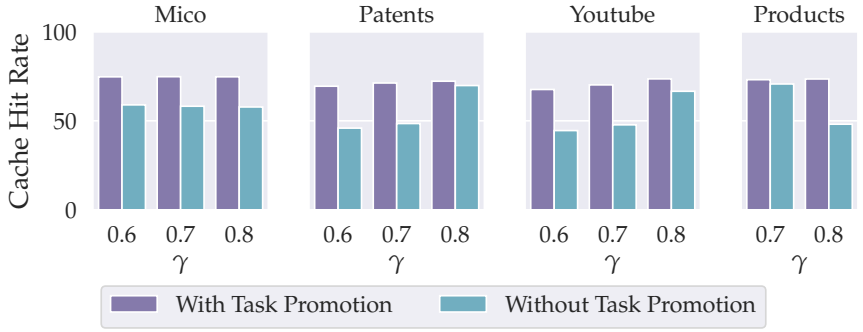
- Generality shown using Quasi-Cliques application w/o maximality constraint
- 2.4-7.2x speedup compared to Peregrine+

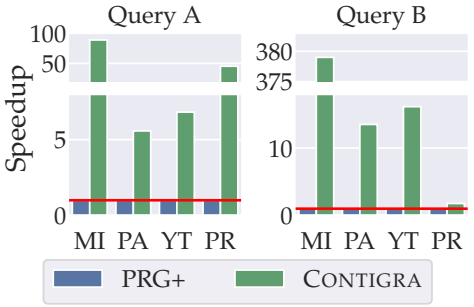
SFU

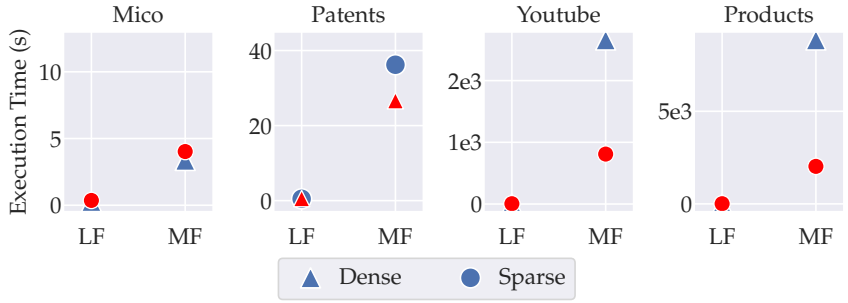
Detailed Experiments

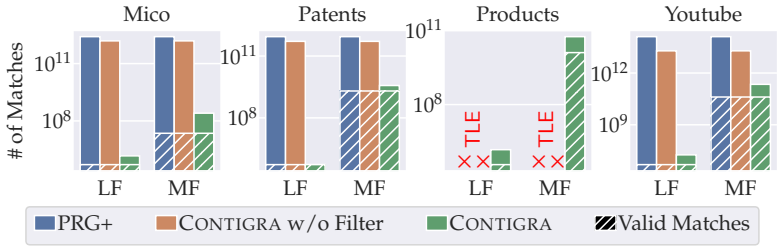


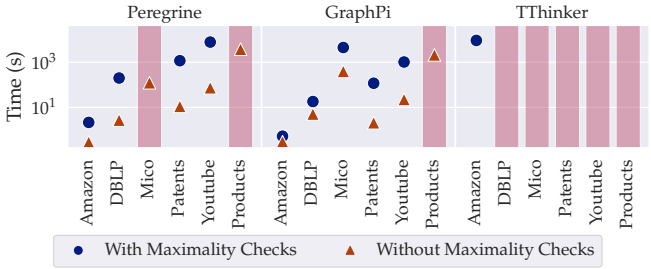


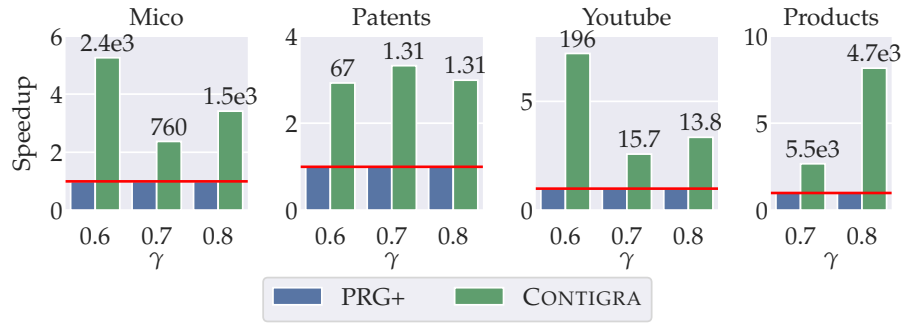


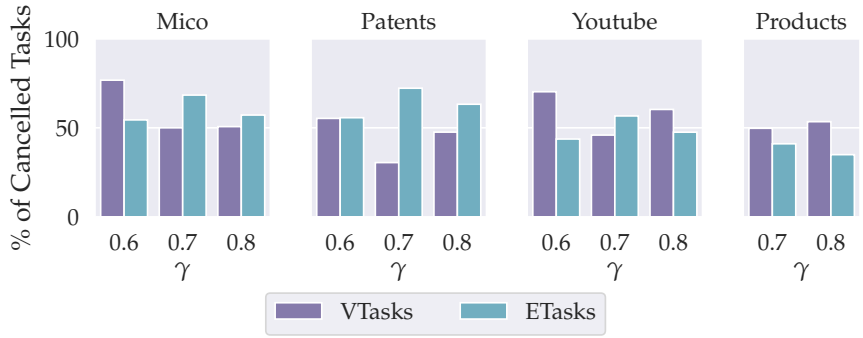












SFU

Conclusion

- First general treatment of containment constraints in graph mining
- Contigra execution model
  - Reduces redundant and unnecessary computations
  - Improves Task-local Cache utilization
- Guided by cross-task dependencies that capture the impact of constraints
- Task management strategies: fusion, promotion, cancellation, skipping, eager filtering
- Scales to complex workloads that existing state-of-the-art cannot handle
- Pushes the boundaries of support for complex graph mining applications

SFU SIMON FRASER UNIVERSITY

NSERC CRSNG