Contigra: Graph Mining with Containment Constraints

Joanna Che  Kasra Jamshidi  Keval Vora

Simon Fraser University

EuroSys 2024
April 23, 2024
Contigra: **Graph Mining with Containment Constraints**
Graph Mining

- Explore subgraphs of interest in large graphs
  - Subgraph isomorphism

![Graph Mining Diagram]

- 4-Cycles
- Chordal 4-Cycle
Graph Mining

- Explore subgraphs of interest in large graphs
  - Subgraph isomorphism
Graph Mining Systems

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

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

```java
void UDF(pattern p, match m) {
    for (vertex v in p) {
        table[p][v].add(m[v]);
    }
}
```
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
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**
Graph Mining with Containment Constraints

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

![Graph Examples]

Maximal Quasi-Cliques
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:** \{red\, blue\, green\}
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 $\mathcal{P}$ with exactly 2 $\mathcal{Q}$

```
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.
Challenge with Containment Constraints

- **Unnecessary** subgraph computation
- **Expensive checking after** each subgraph is explored
**Contigra** for Containment Constraints

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

![Diagram](diagram.png)

```c
void UDF(pattern p, match m) {
  for (vertex v in p) {
    ...
  }
}
```
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
- **Task-local Cache** maintains potential mappings for vertices at each level

![Diagram showing subgraph exploration tasks](image)

- **Exploration Plan**: Subject to $u_1 < u_2$
- **Search Tree**: $c > b$

The diagram illustrates the exploration plan and search tree with vertices labeled from Level 0 to Level 3.
Cross-Task Dependencies from Containment Constraints

- **Successor dependency**

Maximal Quasi-Cliques Application

Task A

<table>
<thead>
<tr>
<th>Level</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task A</td>
<td>c → d → e → d → e → g → h</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Task B

<table>
<thead>
<tr>
<th>Level</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task B</td>
<td>f → g → f → g → f → g → e</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Is Maximal?
Cross-Task Dependencies from Containment Constraints

- **Successor dependency**
- **Predecessor dependency**

---

**Minimal Keyword Search Application**

Task A

Task B

Task C

Level 0 1 2 3 4

Is Minimal?
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**
  - **Task Cancelation**
  - **Task Skipping**
  - **Eager Filtering**

Successor Dependencies

Lateral Dependencies

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

![Diagram of Maximal Quasi-Clique Application]

**Exploration Task**

**Verification Task**

**Validation Task Goal**

 Eliminate unnecessary computation
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

![Diagram](https://example.com/diagram.png)

**Maximal Quasi-Clique Application**

**Exploration Task**
- Level 0
- Level 1
- Level 2
- Level 3
- Level 4

**Verification Task**
- **TASK FUSION**
  - Eliminate unnecessary computation
  - Take a step further to avoid redundant exploration
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

![Graph Diagram]

Maximal Quasi-Clique Application

Level 0 1 2 3 4

Exploration Task
Verification Task

**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**

- **Task Cancelation**
- **Task Skipping**
- **Eager Filtering**
Contigra Execution Model

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

- Task Fusion
- Task Promotion
- **Task Cancelation**
- Task Skipping
- Eager Filtering

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
- Task Cancelation
- Task Skipping
- Eager Filtering

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
- Task Cancelation
- Task Skipping
- Eager Filtering

Successor Dependencies
Lateral Dependencies
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

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

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

<table>
<thead>
<tr>
<th>Application</th>
<th>Baseline</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximal Quasi-Cliques</td>
<td>TThinker</td>
<td>12-41700x</td>
</tr>
<tr>
<td>Nested Subgraph Queries</td>
<td>Peregrine+</td>
<td>5.6-379x</td>
</tr>
<tr>
<td>Keyword Search</td>
<td>Peregrine+</td>
<td>21-16000x</td>
</tr>
<tr>
<td>Quasi-Cliques</td>
<td>Peregrine+</td>
<td>2.4-7.2x</td>
</tr>
</tbody>
</table>
## Performance Summary

<table>
<thead>
<tr>
<th>Application</th>
<th>Baseline</th>
<th>Speedup</th>
<th>Baseline Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximal Quasi-Cliques</td>
<td>TThinker</td>
<td>12-41700x</td>
<td>47%</td>
</tr>
<tr>
<td>Nested Subgraph Queries</td>
<td>Peregrine+</td>
<td>5.6-379x</td>
<td>7%</td>
</tr>
<tr>
<td>Keyword Search</td>
<td>Peregrine+</td>
<td>21-16000x</td>
<td>12%</td>
</tr>
<tr>
<td>Quasi-Cliques</td>
<td>Peregrine+</td>
<td>2.4-7.2x</td>
<td>-</td>
</tr>
</tbody>
</table>
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+
Detailed Experiments
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