Accelerating Graph Mining Systems with Subgraph Morphing

Kasra Jamshidi
Simon Fraser University

Keval Vora
Simon Fraser University

Guoqing Harry Xu
UCLA
Subgraph Morphing

• Peregrine
• AutoZero
• GraphPi
• BiGJoin

100ms overhead – saves 12 hours
Why Graph Mining?

Subgraph morphing.

by 2025, graph technologies will be used in 80% of data and analytics innovations.

"Data analytics] is projected to occupy a market size of USD 329.8 billion by 2030.

- Acumen Research and Consulting

The global data analytics market size [...] is projected to surpass around USD 346.33 billion by 2030.

- Precedence Research

"
What is Graph Mining?

Data Graph

Patterns

User-Defined Function

```java
void UDF(pattern p, match m) {
  count[p] += 1;
}
```
What is Graph Mining?

- Data Graph
- Patterns
- User-Defined Function

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

Data Graph

Patterns

User-Defined Function

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

**Data Graph**

**Patterns**

**User-Defined Function**

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

Data Graph Patterns

User-Defined Function

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

Data Graph

Patterns

User-Defined Function

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

Data Graph

Patterns

User-Defined Function

```java
void UDF(pattern p, match m) {
    for (vertex v in p) {
        table[p][v].add(m[v]);
    }
}
```
What are the bottlenecks?

SUBGRAPH MORPHING
What are the bottlenecks?

- Computing candidate vertices to extend matches
- Filtering duplicate matches
- Generating matches from combinations of candidate vertices
- Calling user application code

Set Operations  Duplicate Avoidance  Materialization  UDF
What are the bottlenecks?

- Counting
- Set operations
- Subgraph morphing

**Diagram:**

<table>
<thead>
<tr>
<th>Time (%)</th>
<th>Set Operations</th>
<th>Materialization</th>
<th>Duplicate Avoidance</th>
<th>UDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.56s</td>
<td>0.61s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.85s</td>
<td>1.96s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.60s</td>
<td>2.77s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.66s</td>
<td>1.85s</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What are the bottlenecks?

- FSM
- Set operations or UDF calls

###-time Bar Chart
- MG: 239s
- MI: 18h

**Legend:**
- Set Operations
- Materialization
- Duplicate Avoidance
- UDF
What are the bottlenecks? Enumeration, set operations, materialization, and UDF calls.
What are the bottlenecks for enumeration, set operations, materialization, and UDF calls? Depends on the pattern.
What are the bottlenecks?
What are the bottlenecks?

MiCo

MAG

Same UDF + same patterns + different data graphs = different relative performance!
What are the bottlenecks?

- So we conclude that performance depends on:
  - Data graph
  - Patterns
  - UDF

What is pattern-centric graph mining?

```cpp
void UDF(pattern p, match m) {
    count[p] += 1;
}
```
Subgraph Morphing

- Replace slow patterns with fast patterns
- Transform results to be consistent with original inputs

Pattern Transformation

Result Transformation
Subgraph Morphing: Intuition

Data Graph

Pattern

Matches

SUBGRAPH

MORPHING
Subgraph Morphing: Intuition

Data Graph

Pattern

Matches
Subgraph Morphing: Intuition
Subgraph Morphing: Intuition

Data Graph Pattern Matches
Subgraph Morphing: Intuition
Subgraph Morphing: Intuition

Data Graph Pattern

MATCHES
Subgraph Morphing: Intuition

Data Graph

Pattern

Matches
Subgraph Morphing: Intuition

Data Graph Pattern Matches
Subgraph Morphing: Intuition

Diverse Data Graph Pattern Matches

- Matches:
  - 3
  - +
  - SUBGRAPH MORPHING

Graphs:

- Initial graph
- Simple graphs
- More complex graphs
Subgraph Morphing: Theory
Subgraph Morphing: Theory

Data Graph

Pattern

$\mathcal{G} \cup \mathcal{H}$
Subgraph Morphing: Theory

Data Graph

Pattern Matches

\[ S \cup S \cup S \]
Subgraph Morphing: Theory

Data Graph

SUBGRAPH MORPHING

15
Subgraph Morphing: Theory

Data Graph

\( \phi \)
Subgraph Morphing: Theory

Data Graph

Pattern

$S_{UBGRAPH} \cup \cup$
Subgraph Morphing: Theory

\[ \text{Data Graph} \cup \phi \cup \phi \cup \phi \]
Subgraph Morphing: Theory

Data Graph

$\bigcup_\phi$ $\bigcup_\phi$ $\bigcup_\phi$
Subgraph Morphing: Theory

Data Graph: Patterns Matches

$\phi \oplus \phi \oplus \phi$
Subgraph Morphing

- Replace slow patterns with fast patterns
- Transform results to be consistent with original inputs
Pattern Transformation

= 21
\[ \text{SUBGRAPH MORPHING} \]
Pattern Transformation

\[ \begin{align*}
\text{SUBGRAPH} & \quad \text{MORPHING} \\
= & \quad -4 \quad -12 \\
\end{align*} \]
Pattern Transformation

\[
\begin{align*}
\text{SUBGRAPH MORPHING} & \quad 21
\end{align*}
\]
Pattern Transformation

\[
\begin{align*}
\text{SUBGRAPH} & \quad = \quad \text{SUBGRAPH} + 4 \times \text{SUBGRAPH} - 6 \\
\text{SUBGRAPH} & \quad = \quad \text{SUBGRAPH} - 6
\end{align*}
\]
\[ \text{Pattern Transformation} = \text{SUBGRAPH} \]
\[ \begin{align*}
&= \bigcirc \bigcirc + 4 \bigcirc \bigcirc - 6 \\
&= \bigcirc \bigcirc + 4 \bigcirc \bigcirc + 12 \\
&= \bigcirc \bigcirc + 2 \bigcirc \bigcirc + 2
\end{align*} \]
Pattern Transformation

- Explore combinations of different patterns
- Choose an efficient combination

SUBGRAPH MORPHING
Pattern Transformation
• Explore combinations of different patterns
• Choose an efficient combination

SUBGRAPH MORPHING
Pattern Transformation: Estimating Cost

- Tap into the underyling system's cost-based optimizer
- Model data graph
- Model pattern matching as nested loops
- Compute work done across all loops
Pattern Transformation: Estimating Cost

- Tap into the underlying system's cost-based optimizer
- Account for UDF costs (i.e., per-match overhead)
Pattern Transformation

• Explore combinations of different patterns
• Choose an efficient combination
Input Pattern

S-DAG

Pattern Transformation

S

UBGRAPH

M

ORPHING

25
Pattern Transformation

Input Pattern $s$

S-DAG

Whichever variant has lower cost $S$$UBGRAPH$$M$$ORPHING$ 25
Pattern Transformation

Input Pattern $s$

$S$-DAG

Alternative Pattern $s$

$S$UBGRAPH $M$

MORPHING

**Cost:** 700
Pattern Transformation

Input Pattern

S-DAG

SUBGRAPH MORPHING

V: 500
E: 5

V: 200
E: 50

Alternative Patterns

Cost: 700
Pattern Transformation

Input Patterns

S-DAG

V: 500
E: 5

50
150

V: 200
E: 50

25

Alternative Patterns

Cost: 700
Pattern Transformation

Input Pattern

S-DAG

V > + S

SUBGRAPH MORPHING

V: 500

E: 5

Alternative Pattern

Cost: 700
Pattern Transformation

Input Pattern

S-DAG

\[ S \rightarrow + \]

\( S \)UB\( G \)RAF\( M \)ORPHING

50

V: 500
E: 5

50

150

200

50

25

25

Alternative Pattern

Cost: 700
Pattern Transformation

Input

Pattern s

S-DAG

SUBGRAPH MORPHING

V: 500
E: 5

V: 200
E: 50

Alternative Pattern s

Cost: 700
Pattern Transformation

Input Pattern

S-DAG

V > + S

SUBGRAPH MORPHING

V: 500 E: 50

V: 200 E: 5

Alternative Pattern

Cost: 700
Pattern Transformation

Input Pattern

S-DAG

SUBGRAPH MORPHING

Alternative Patterns

Cost: 700

> + ?
Pattern Transformation

Input Pattern

S-DAG

V: 500
E: 5

Alternative Pattern Cost: 700

Set E to 0, SUBGRAPH MORPHING

V: 200
E: 50

\( > \) + \( ? \) Set \( , \) to 0
Pattern Transformation

Input Pattern $s$

$S$-DAG

$V > + ?$ Set $E$ to 0,

Alternative Pattern

Cost: 455
Pattern Transformation

Input Pattern

S-DAG

SUBGRAPH MORPHING

V: 500
E: 0

V: 200
E: 50

Alternative Pattern

Cost: 455
Pattern Transformation

Input

Pattern $s$

S-DAG $V \rightarrow E$

SUBGRAPH MORPHING

0 0

V: 500
E: 0

V: 200
E: 50

0

Cost: 455

Alternative Pattern $s$

> ?
Pattern Transformation

Input Pattern

S-DAG

V: 500
E: 0

V: 200
E: 0

Alternative Pattern

Cost: 305

> ?
Pattern Transformation

Input Pattern

Alternative Pattern

SUBGRAPH MORPHING
Result Transformation

SUBGRAPH

MORPHING
Result Transformation

• On-the-Fly
• Match alternative patterns
Result Transformation

• On-the-Fly
• Match alternative patterns
• Permute each match according to $\phi$ before applying UDF
Result Transformation

• On-the-Fly
• Match alternative patterns
• Permute each match according to $\phi$ before applying UDF

...
Result Transformation

• On-the-Fly
• Match alternative patterns
• Permute each match according to $\phi$ before applying UDF

Post-Matching

• Match alternative patterns and apply UDF normally

$\phi$

SUBGRAPH MORPHING
Result Transformation

• On-the-Fly
• Match alternative patterns
• Permute each match according to \( \phi \) before applying UDF

• Post-Matching
• Match alternative patterns and apply UDF normally
• Permute aggregation keys according to \( \phi \)

\( \phi \)
Result Transformation

- On-the-Fly
- Match alternative patterns
- Permute each match according to $\phi$ before applying UDF

- Post-Matching
- Match alternative patterns and apply UDF normally
- Permute aggregation keys according to $\phi$
- Accumulate results

$S$UBGRAPH $M$ORPHING
Subgraph Morphing

- Replace slow patterns with fast patterns
- Adjust results to be consistent with original inputs

Pattern Transformation

Graph Mining System

SUBGRAPH MORMPHING
Evaluation

• 5 datasets

SUBGRAPH

MiCo \(|E| = 1M\)

Products \(|E| = 61M\)

MAG \(|E| = 5.4M\)

Orkut \(|E| = 117M\)

Friendster \(|E| = 1.8B\)
Evaluation

- 5 datasets
- 4 systems
  - Peregrine [EuroSys '20]
  - AutoZero [SOSP '19/OSR '21]
  - GraphPi [SC '20]
  - BiGJoin [VLDB '18]

 Subgraph

Morphing

30
• 5 datasets
• 4 systems
• 21 pattern sets
  • Different combinations of patterns
  • Different variants of patterns
  • \{3,4\}-FSM
  • \{3,4,5\}-MC
• 5 datasets
• 4 systems
• 21 pattern sets
• Execution time
• Bottleneck analysis

SUBGRAPH MORPHING

30
Evaluation Highlights

- Best case motif-counting

Peregrine AutoZero SUBGRAPH MORPHING

34x

31
Evaluation: Highlights

- Best case: motif-counting
- Worst case: single patterns

### Speedup

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIMGPRKFR</td>
<td>1.16</td>
<td>0.75</td>
<td>0.44</td>
<td>0.75</td>
<td>0.85</td>
</tr>
<tr>
<td>p₁</td>
<td>52.56</td>
<td>177.79</td>
<td>44.11</td>
<td>156.99</td>
<td>691.32</td>
</tr>
<tr>
<td>p₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₁, p₂</td>
<td>65.62</td>
<td>0.75</td>
<td>0.75</td>
<td>114.89</td>
<td>1060.82</td>
</tr>
<tr>
<td>p₃</td>
<td>12.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₄</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₅</td>
<td>4305.19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₆</td>
<td>9489.79</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₇</td>
<td>31.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₈</td>
<td>9.56</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₉</td>
<td>19045.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₁₀</td>
<td>27014.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₁₁</td>
<td>88.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₁₂</td>
<td>4759.55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₁₃</td>
<td>5059.92</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p₁₄</td>
<td>3526.59</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluation: Highlights

- Best case: motif-counting
- Worst case: single patterns
- Large pattern enumeration

34x 24x 7x

GraphPi Peregrine SUBGRAPH MORPHING 31
Evaluation: Highlights

- Best case: motif-counting
- Worst case: single patterns
- Large pattern enumeration
- On-the-fly conversion

![Speedup Diagram]

<table>
<thead>
<tr>
<th></th>
<th>MI</th>
<th>MIPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 PE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 PE</td>
<td>43.89</td>
<td></td>
</tr>
<tr>
<td>9 PE</td>
<td>23776.28</td>
<td></td>
</tr>
</tbody>
</table>
Evaluation: Highlights

- Best case \(\rightarrow\) motif-counting
- Worst case \(\rightarrow\) single patterns
- Large pattern enumeration
- On-the-fly conversion
Conclusion

• Graph mining performance bottlenecks vary wildly between different application workloads.
• We address all bottlenecks automatically, by exploiting inherent performance differences across patterns.
• Subgraph Morphing integrates into any pattern-based system with any graph mining application.
• Massive speedups with millisecond overheads.