Accelerating Graph Mining Systems with Subgraph Morphing

Kasra Jamshidi
Simon Fraser University

Guoqing Harry Xu
UCLA

Keval Vora
Simon Fraser University

Subgraph Morphing

- Peregrine 34x ⬆
- AutoZero 10x ⬆
- GraphPi 18x ⬆
- BiGJoin 13x ⬆

100ms overhead – saves 12 hours

Why Graph Mining?

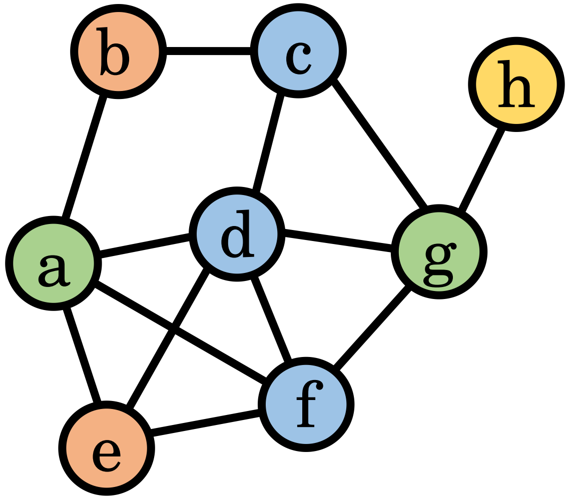
“ *[…] by 2025, graph technologies will be used in* ***80%*** *of data and analytics innovations.*

Gartner

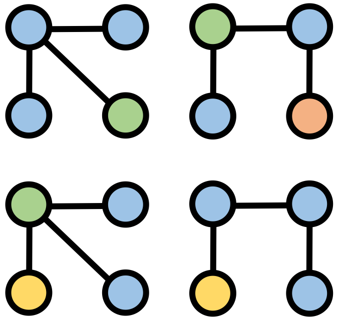
“ *The global data analytics market size […] is projected to surpass around* ***USD 346.33 billion*** *by 2030.*
- Precedence Research

“ *[Data analytics] is projected to occupy a market size of* ***USD 329.8 billion*** *by 2030.*
- Acumen Research and Consulting

What is Graph Mining?



Data Graph

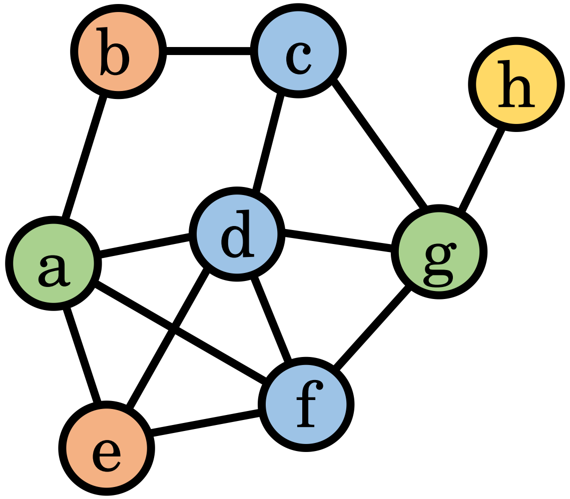


Patterns

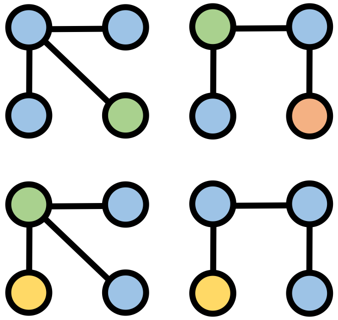
```
void UDF(pattern p, match m) {
  count[p] += 1;
}
```

User-Defined Function

What is Graph Mining?



Data Graph

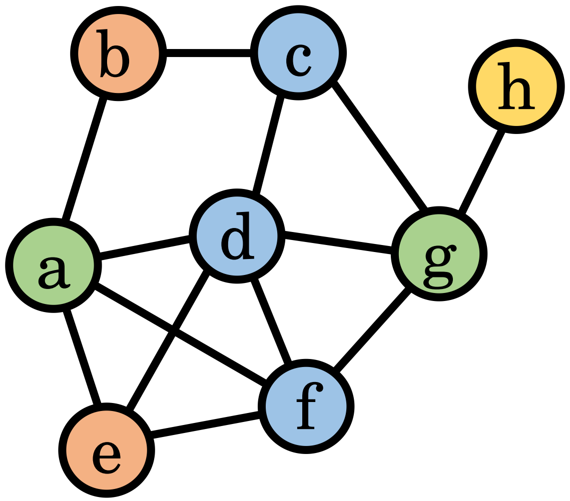


Patterns

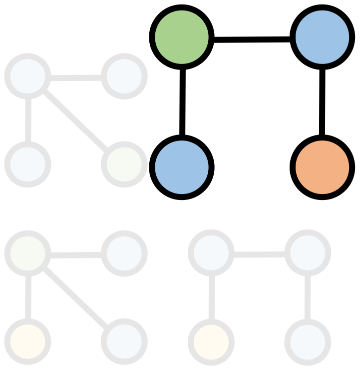
```
void UDF(pattern p, match m) {
  for (vertex v in p) {
    table[p][v].add(m[v]);
  }
}
```

User-Defined Function

What is Graph Mining?



Data Graph

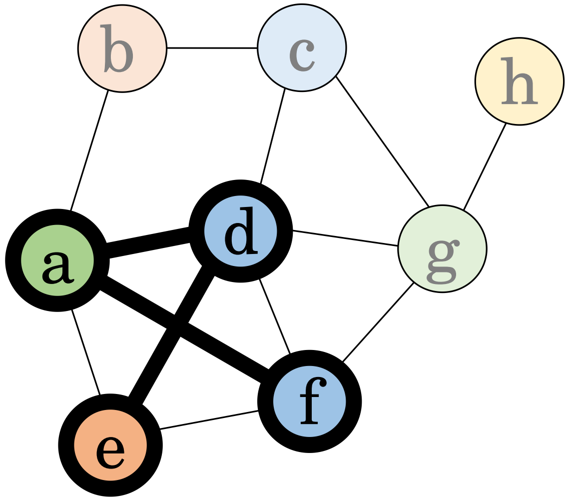


Patterns

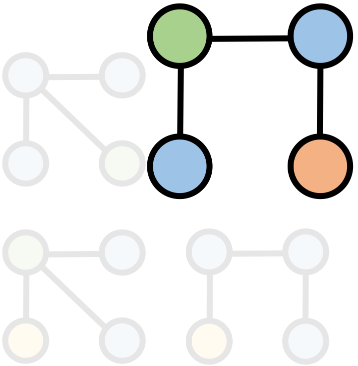
```
void UDF(pattern p, match m) {
  for (vertex v in p) {
    table[p][v].add(m[v]);
  }
}
```

User-Defined Function

What is Graph Mining?



Data Graph

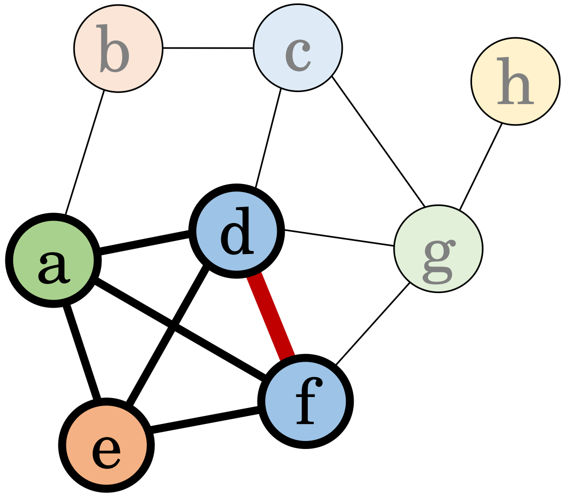


Patterns

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

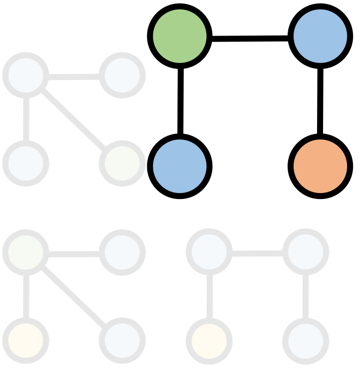
User-Defined Function

What is Graph Mining?



Data Graph

Edge-Induced

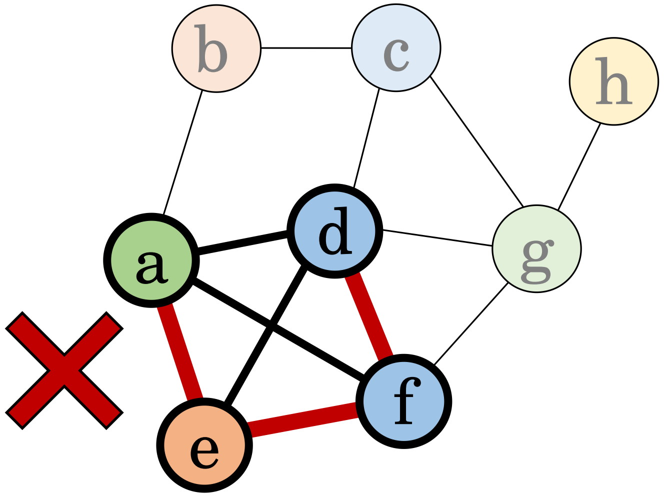


Patterns

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

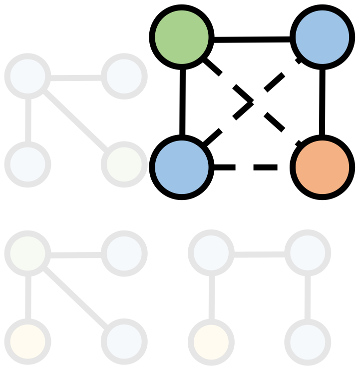
User-Defined Function

What is Graph Mining?



Data Graph

Vertex-Induced



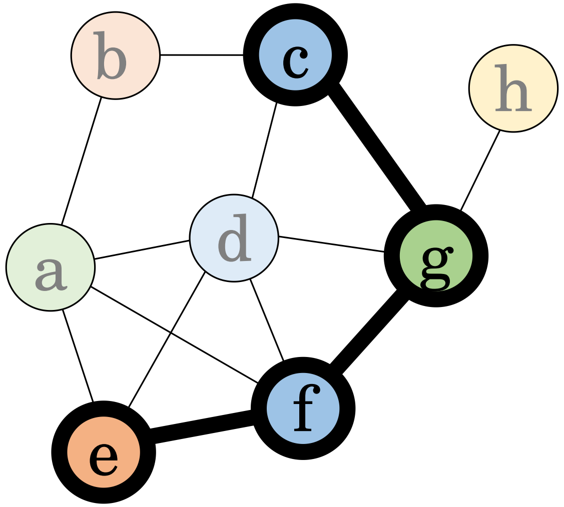
Patterns

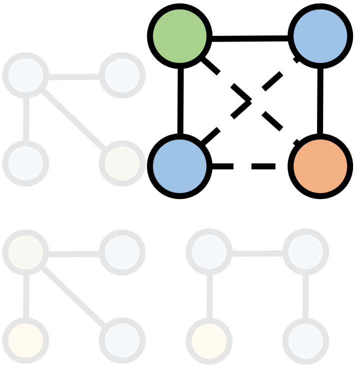
```
void UDF(pattern p, match m) {
  for (vertex v in p) {
    table[p][v].add(m[v]);
  }
}
```

User-Defined Function

What is Graph Mining?

Vertex-Induced





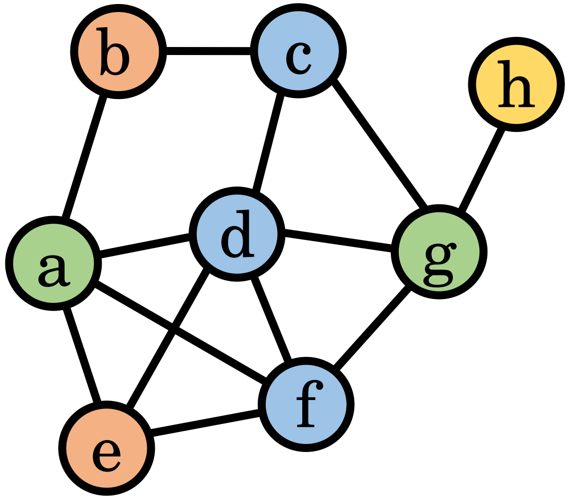
```
void UDF(pattern p, match m) {
  for (vertex v in p) {
    table[p][v].add(m[v]);
  }
}
```

Data Graph

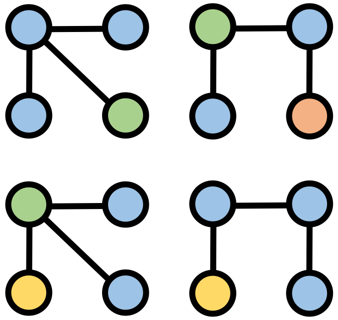
Patterns

User-Defined Function

What is Graph Mining?



Data Graph



Patterns

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

User-Defined Function

What are the bottlenecks?

What are the bottlenecks?

Calling user application code

Filtering duplicate matches

Set Operations | Duplicate Avoidance | Materialization | UDF

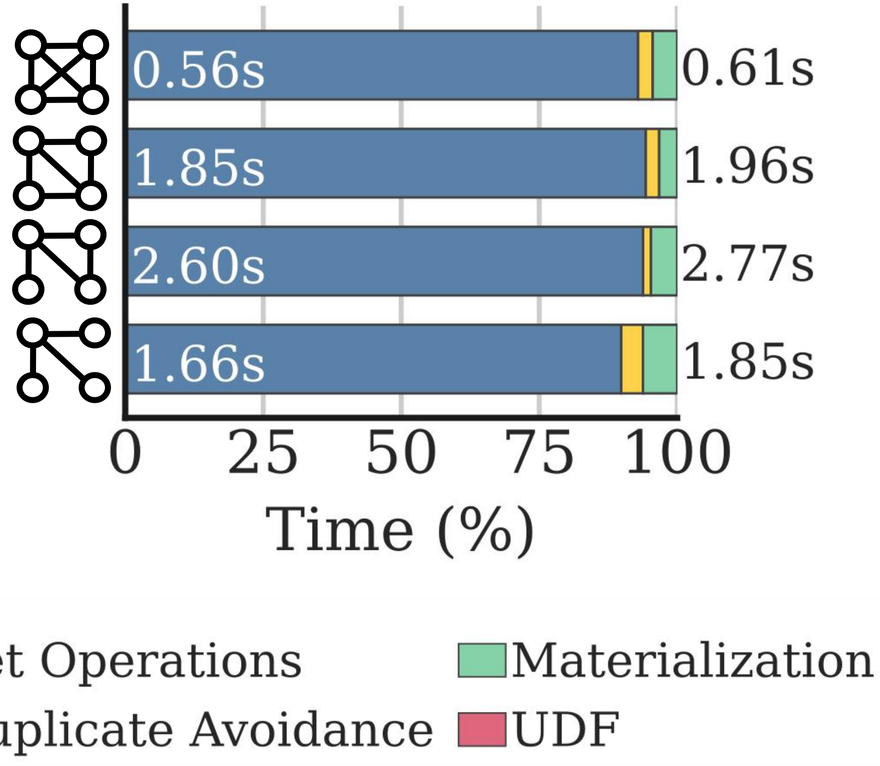
Computing candidate vertices to extend matches

Generating matches from combinations of candidate vertices

What are the bottlenecks?

Counting

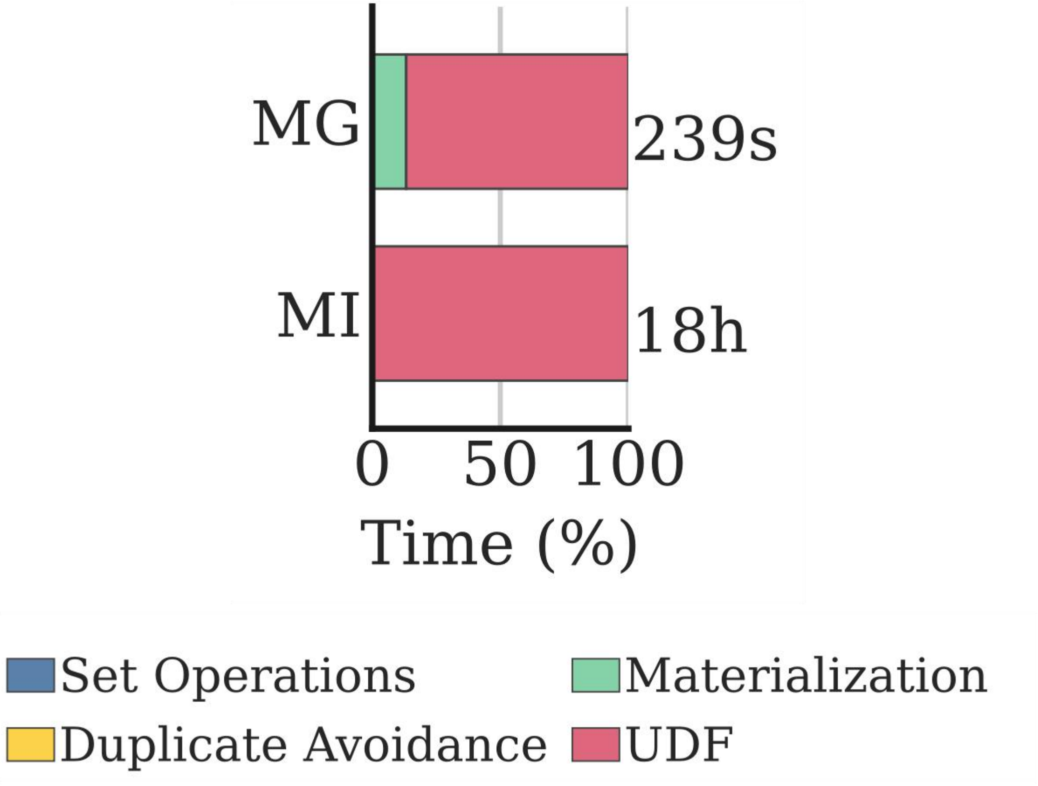
Set operations



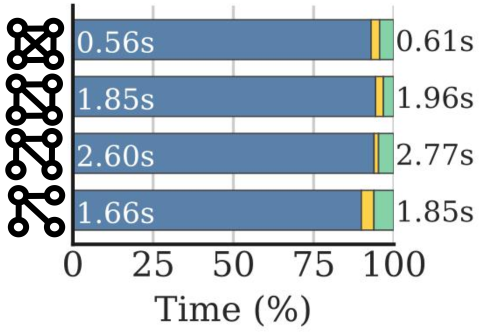
What are the bottlenecks?

Set operations or
UDF calls

FSM



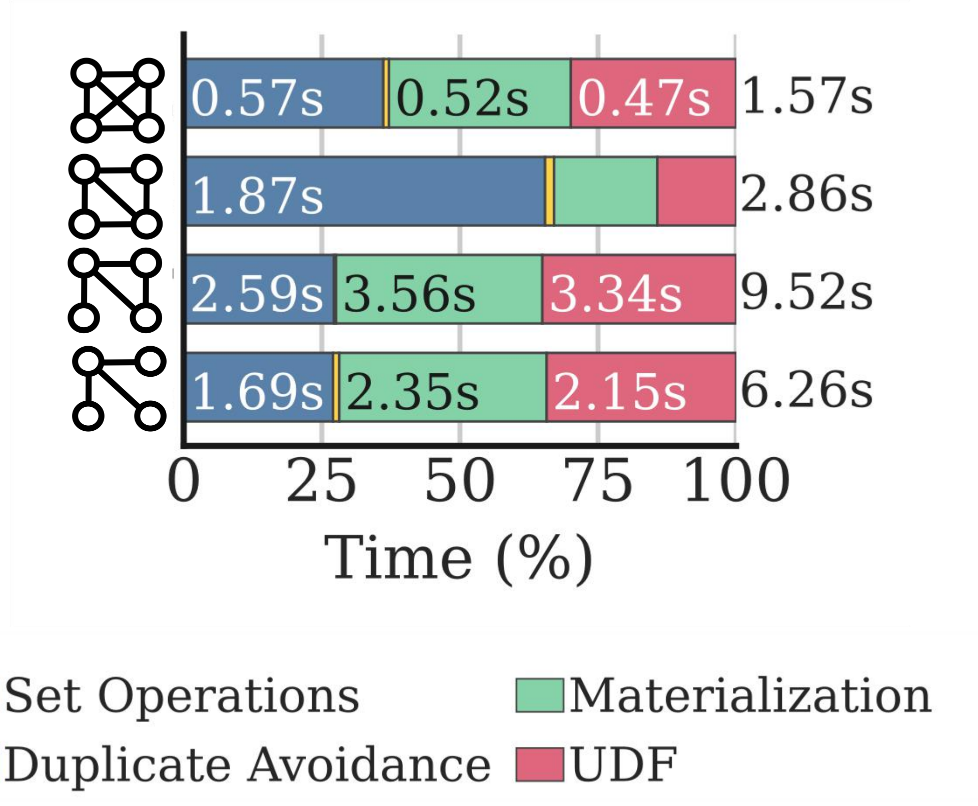
Counting



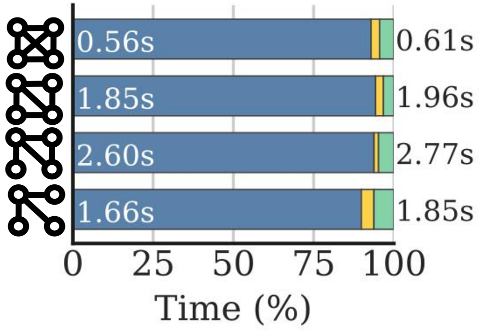
What are the bottlenecks?

Set operations,
materialization,
and UDF calls?

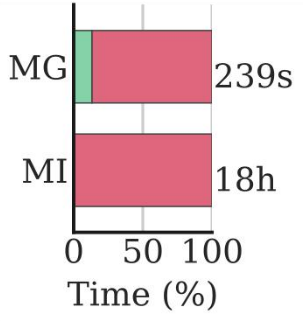
Enumeration



Counting



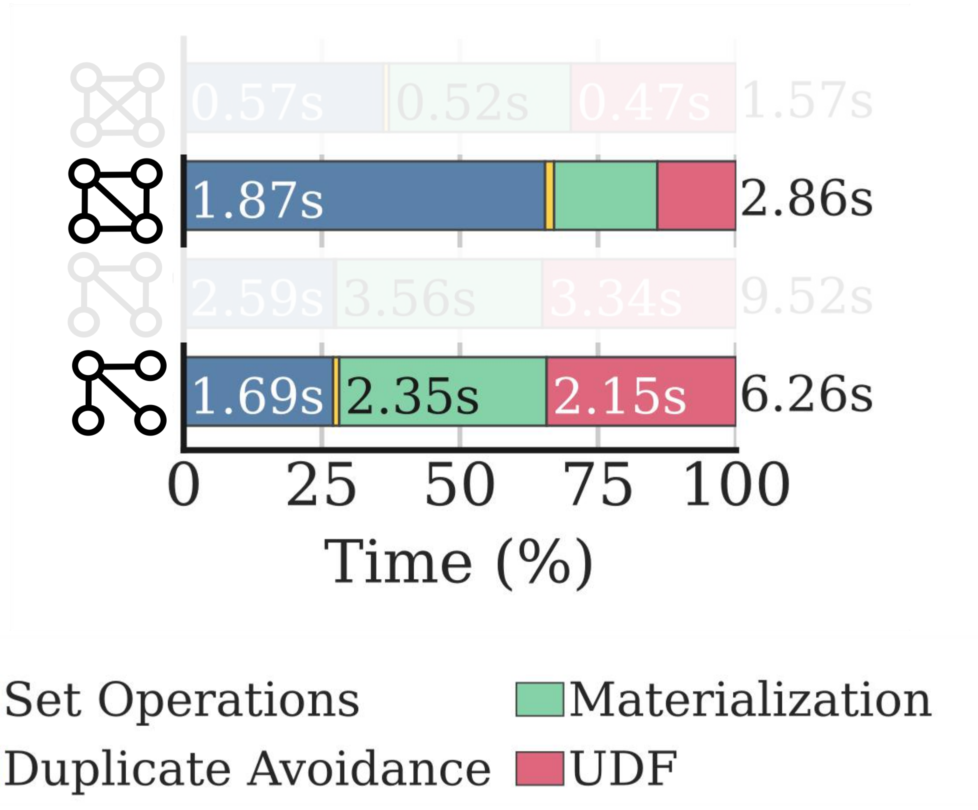
FSM



What are the bottlenecks?

Set operations, materialization, *and* UDF calls?
Depends on the pattern?

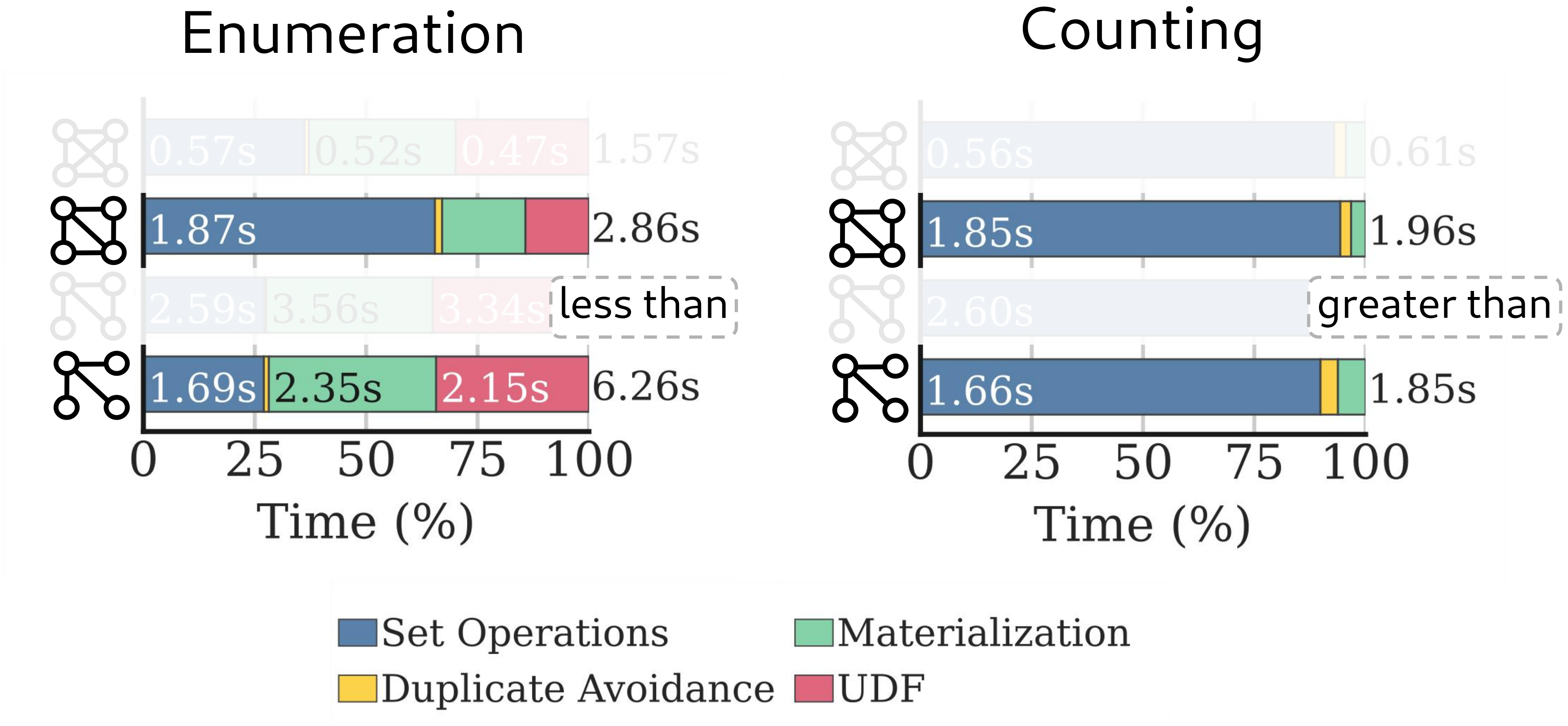
Enumeration



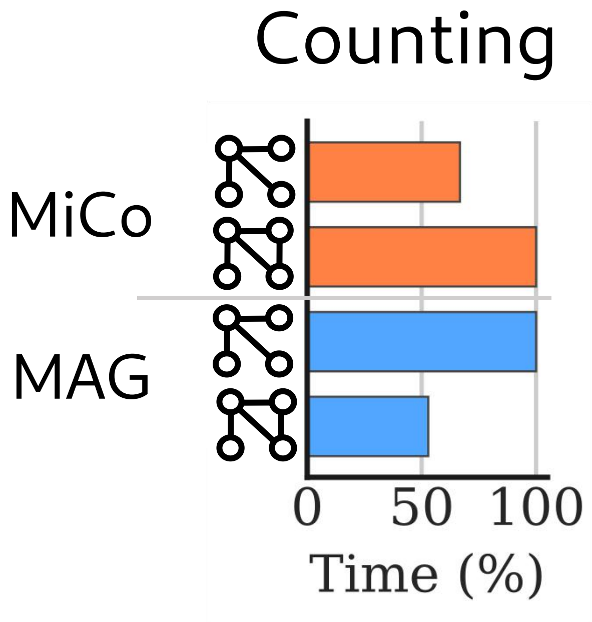
Counting

FSM

What are the bottlenecks?



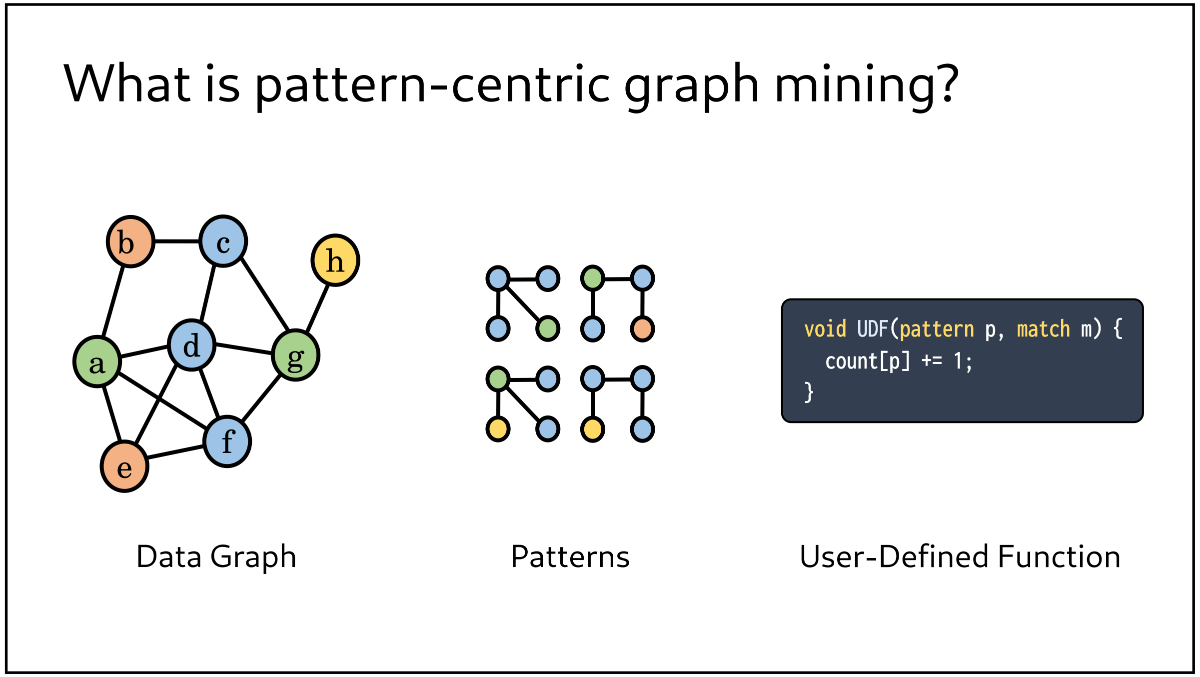
What are the bottlenecks?



Same UDF + same patterns + different data graphs
= different relative performance!

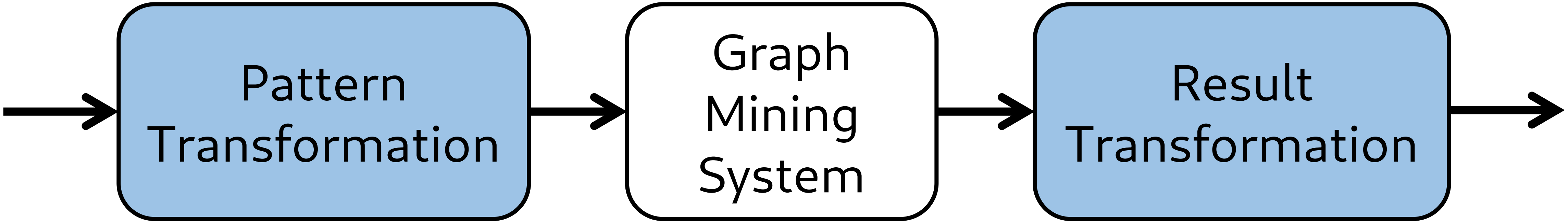
What are the bottlenecks?

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

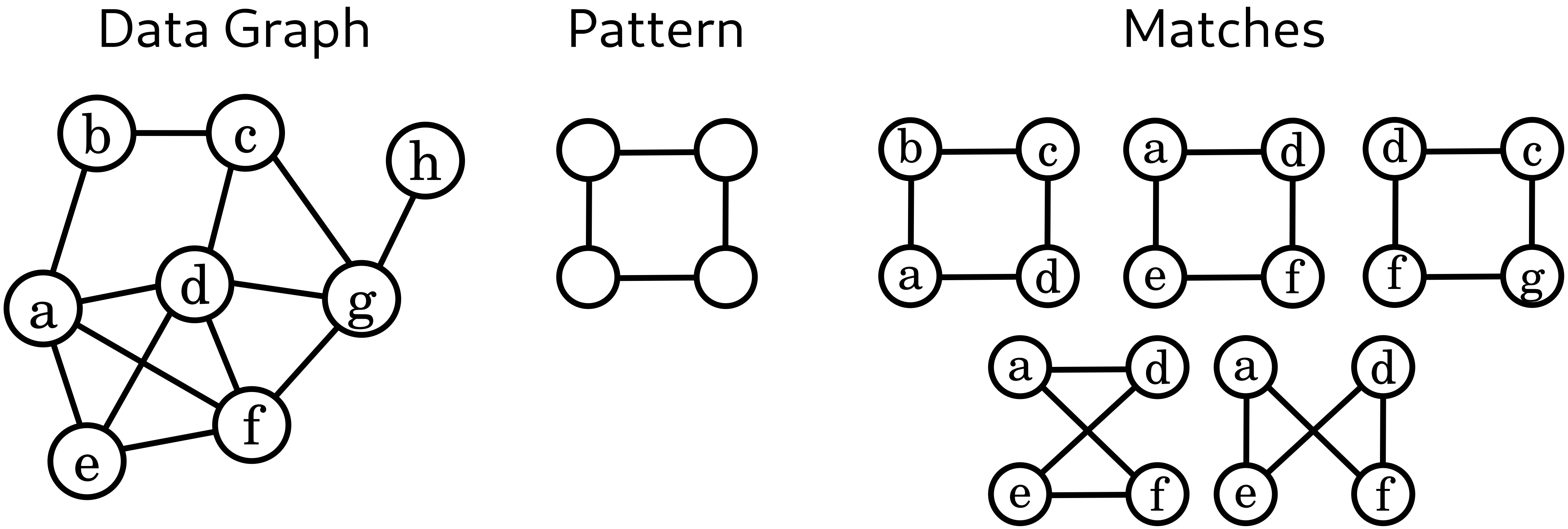


Subgraph Morphing

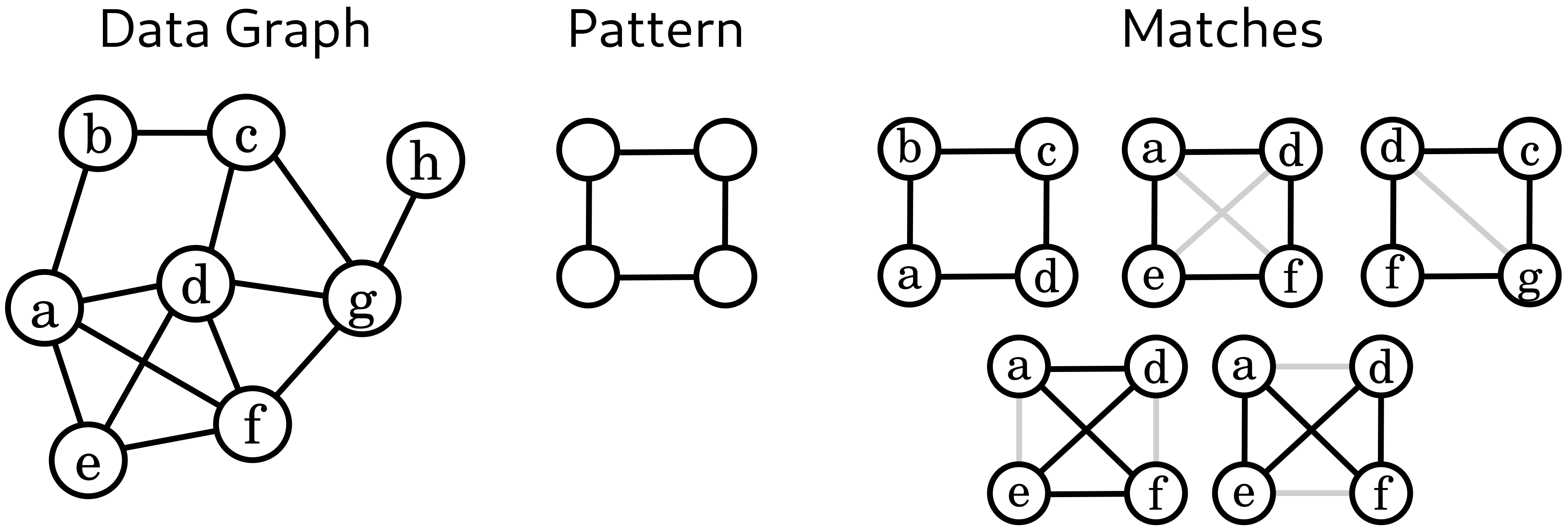
- Replace slow patterns with fast patterns
- Transform results to be consistent with original inputs



Subgraph Morphing: Intuition

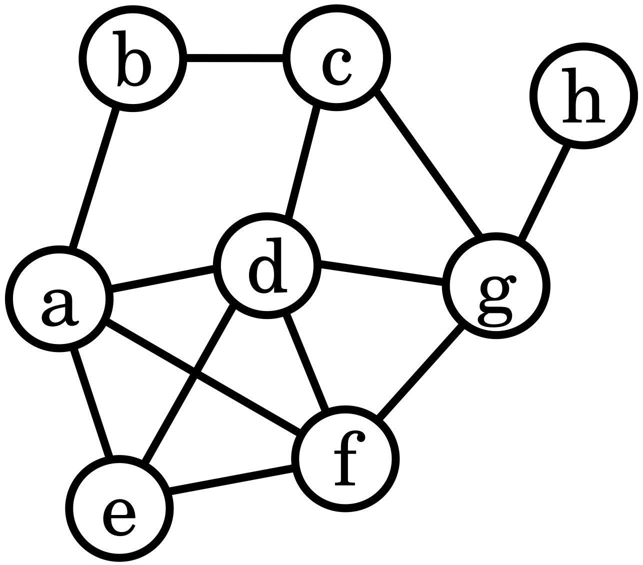


Subgraph Morphing: Intuition

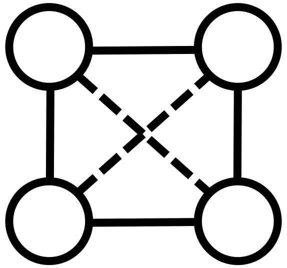


Subgraph Morphing: Intuition

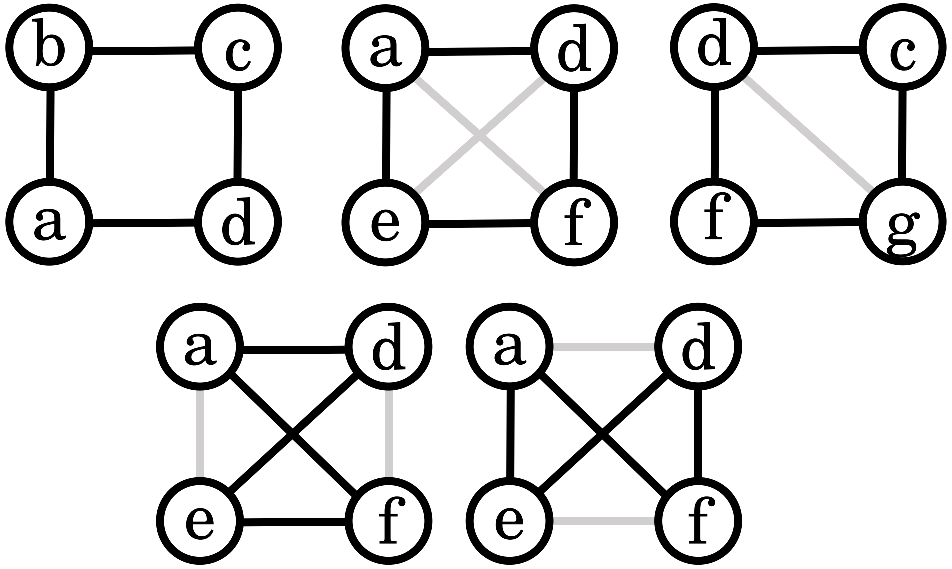
Data Graph



Pattern

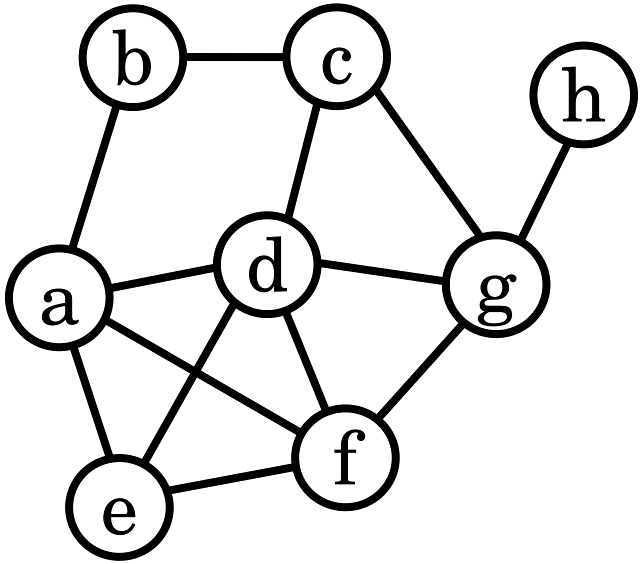


Matches

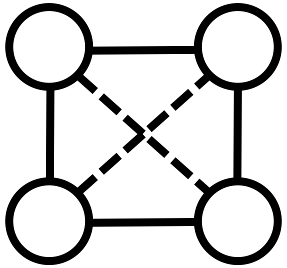


Subgraph Morphing: Intuition

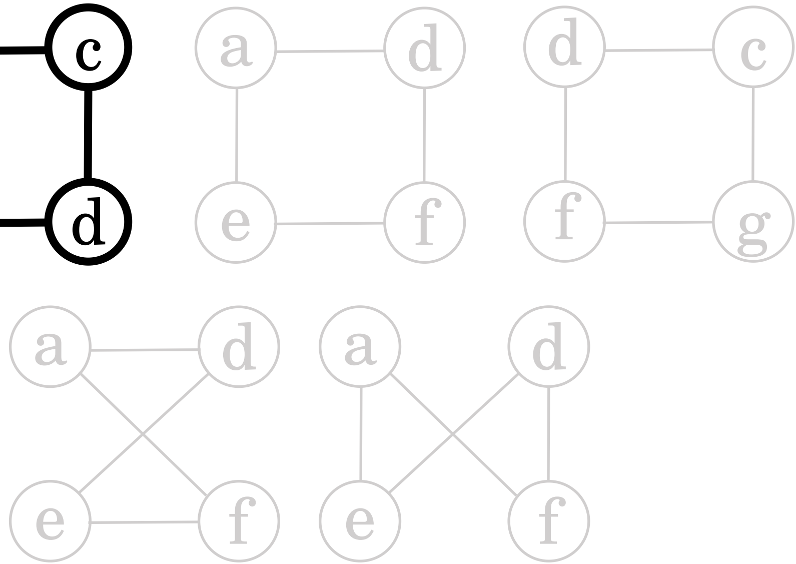
Data Graph



Pattern

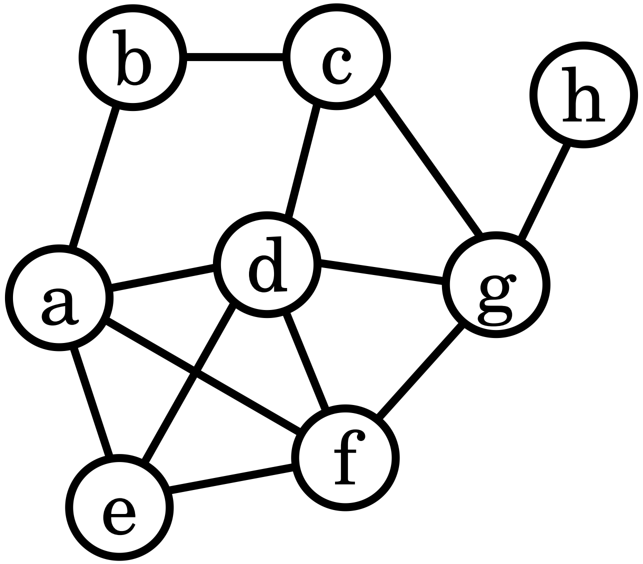


Matches

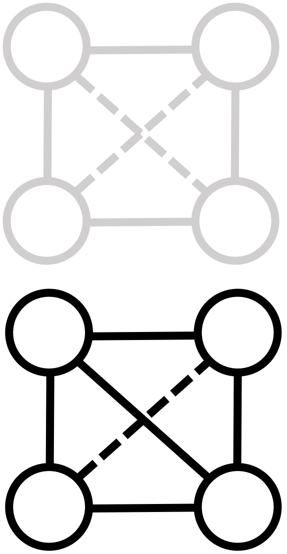


Subgraph Morphing: Intuition

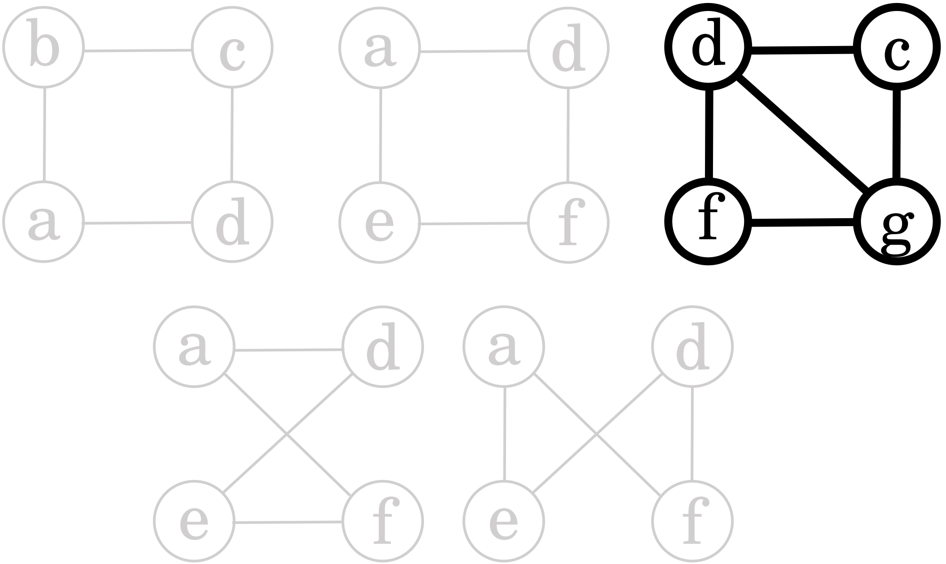
Data Graph



Pattern

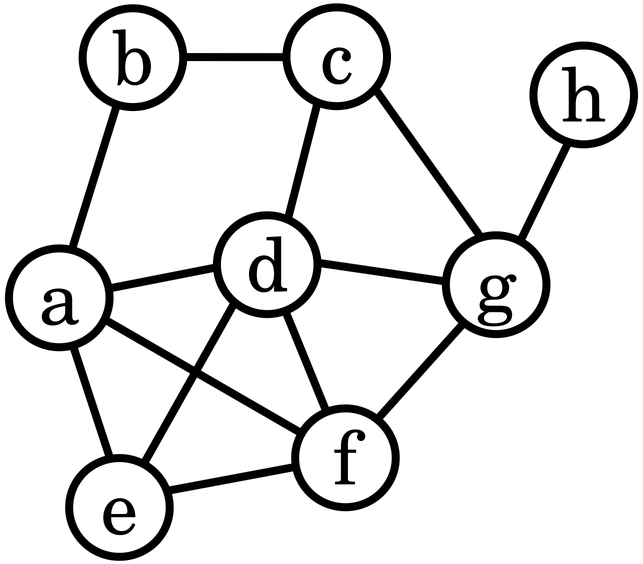


Matches

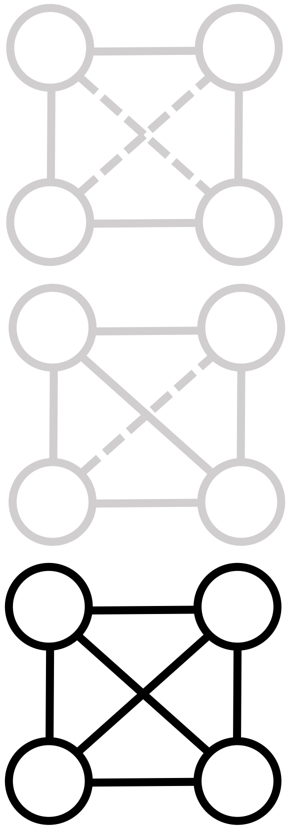


Subgraph Morphing: Intuition

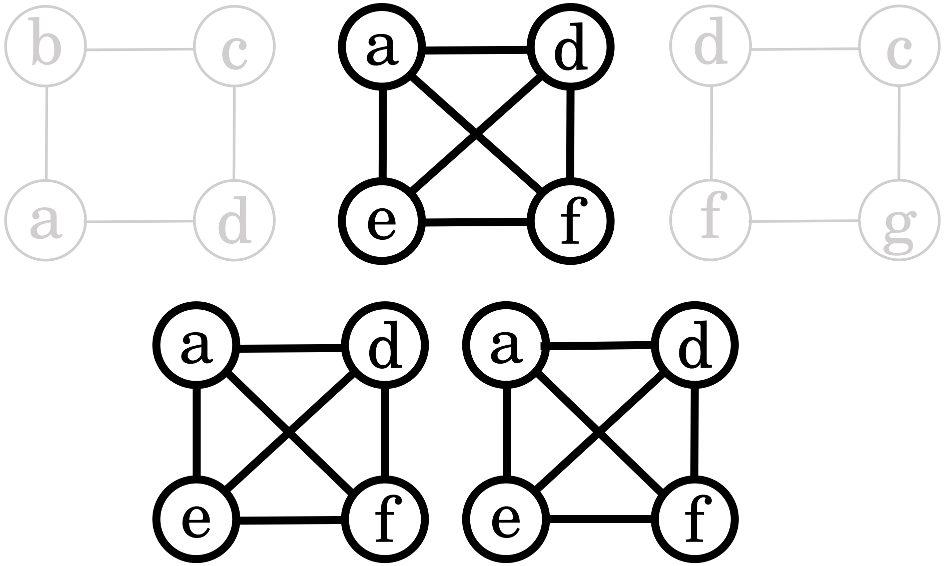
Data Graph



Pattern

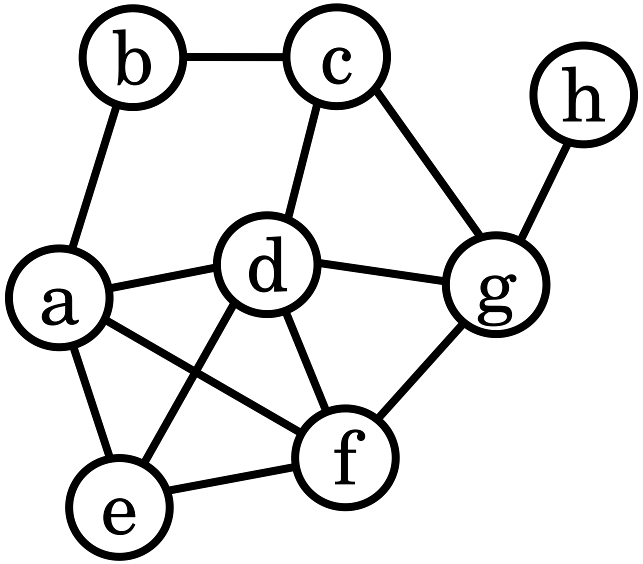


Matches

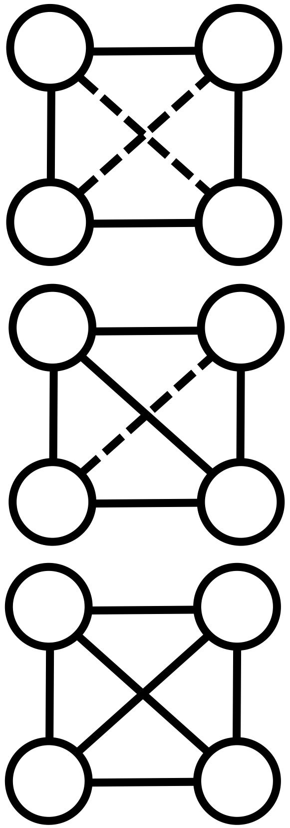


Subgraph Morphing: Intuition

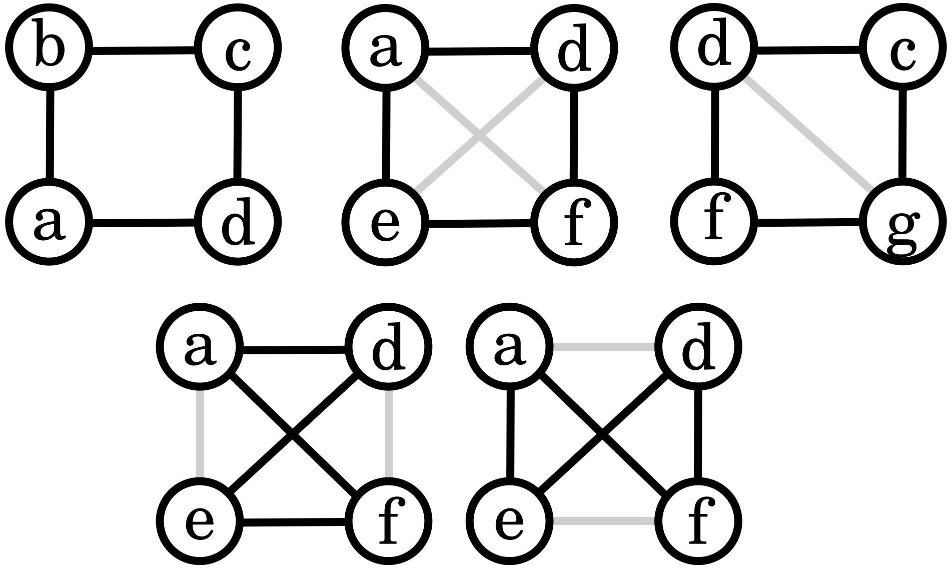
Data Graph



Pattern

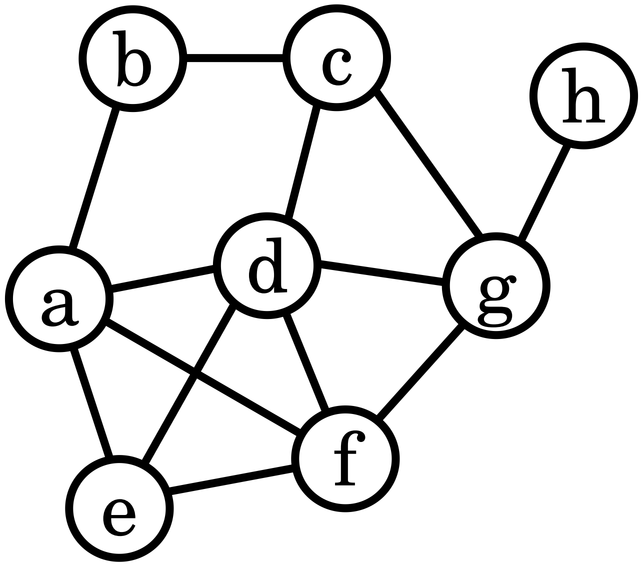


Matches

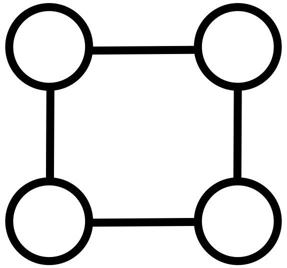


Subgraph Morphing: Intuition

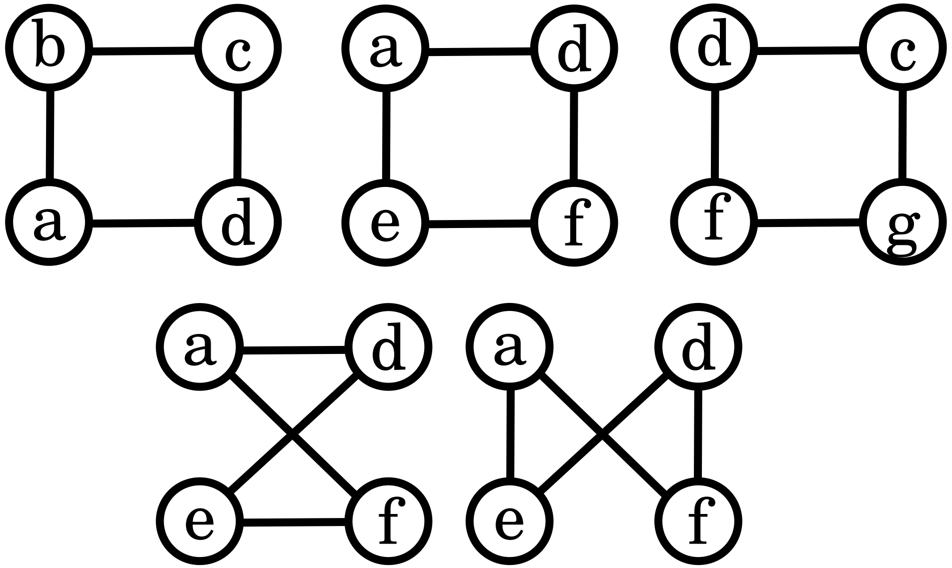
Data Graph



Pattern

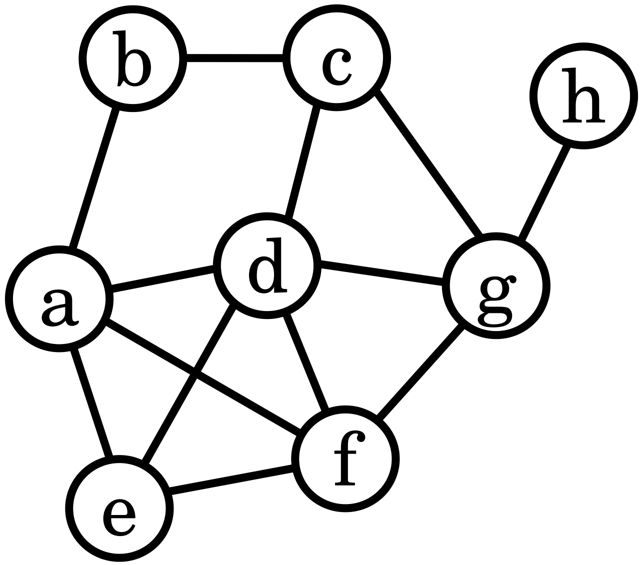


Matches

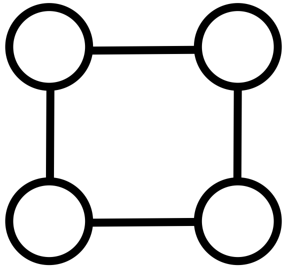


Subgraph Morphing: Intuition

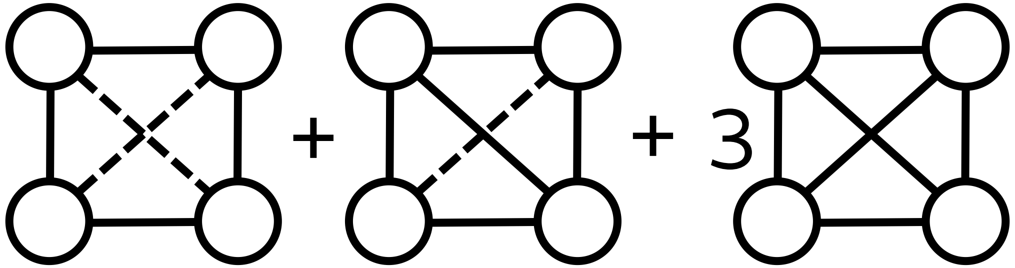
Data Graph



Pattern

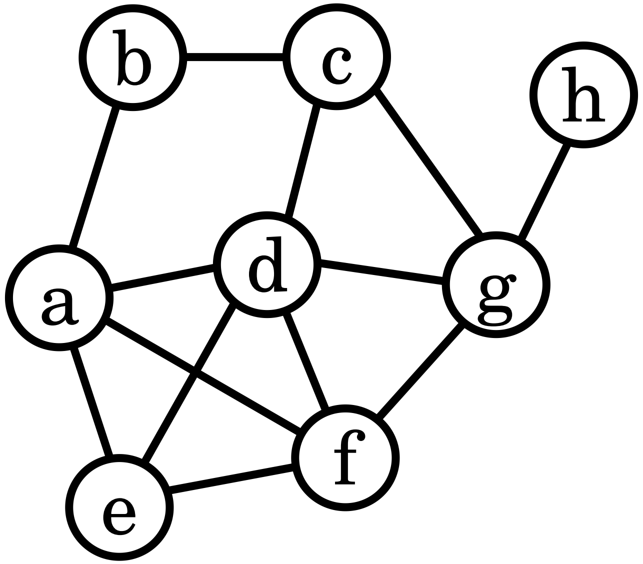


|Matches|

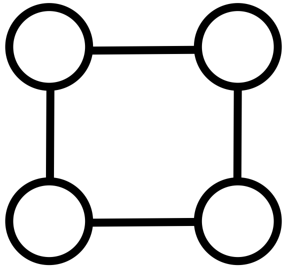


Subgraph Morphing: Theory

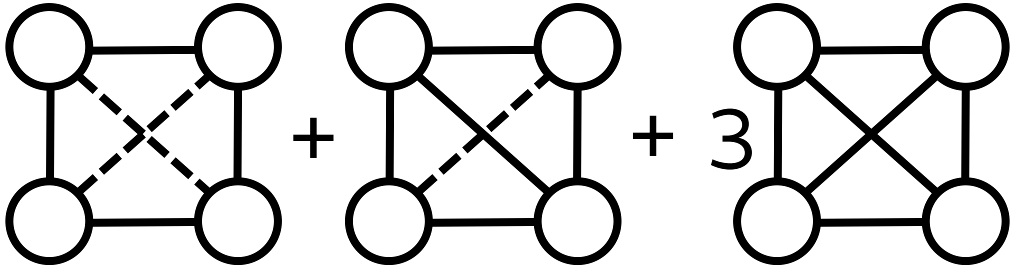
Data Graph



Pattern

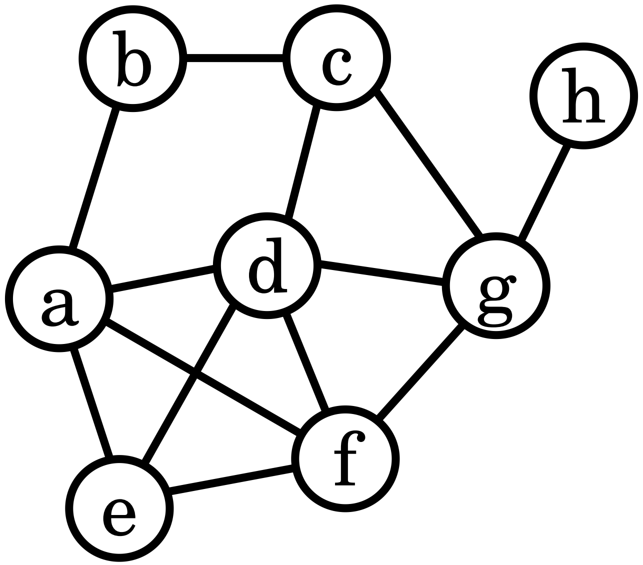


|Matches|

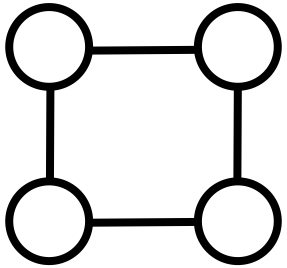


Subgraph Morphing: Theory

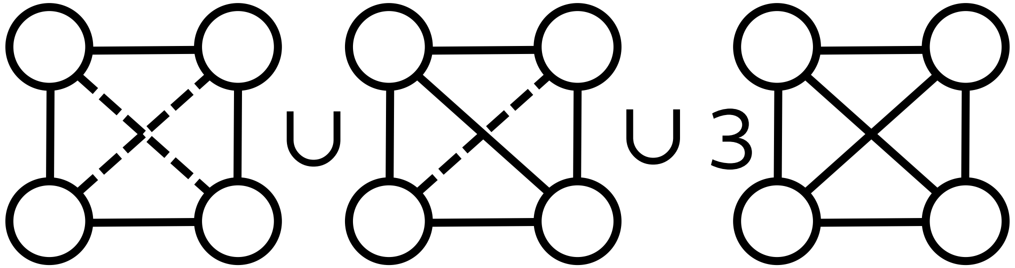
Data Graph



Pattern

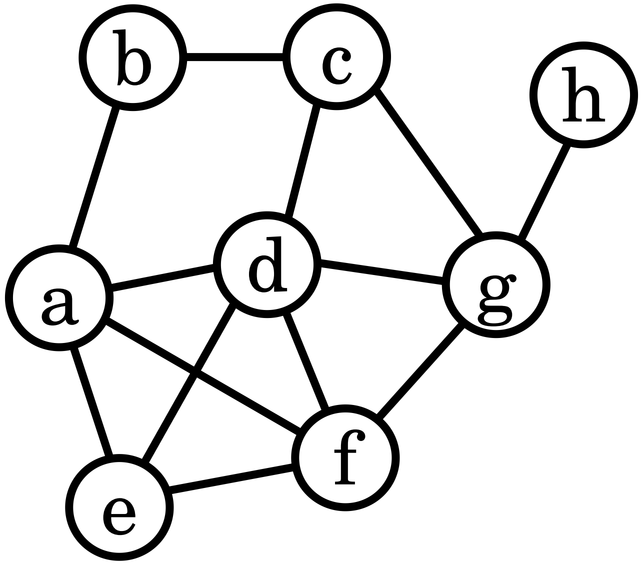


Matches

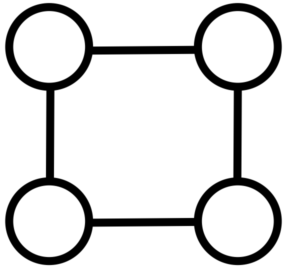


Subgraph Morphing: Theory

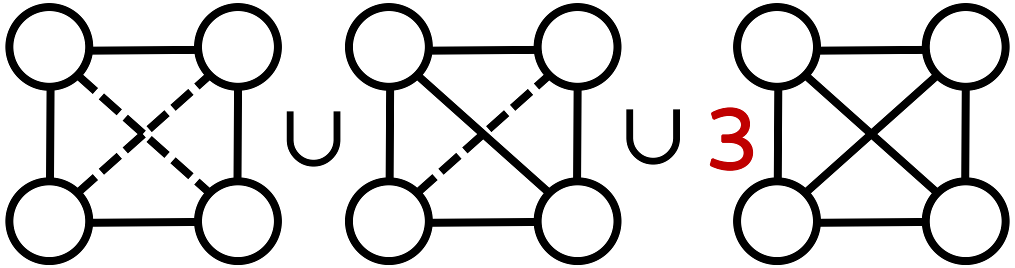
Data Graph



Pattern

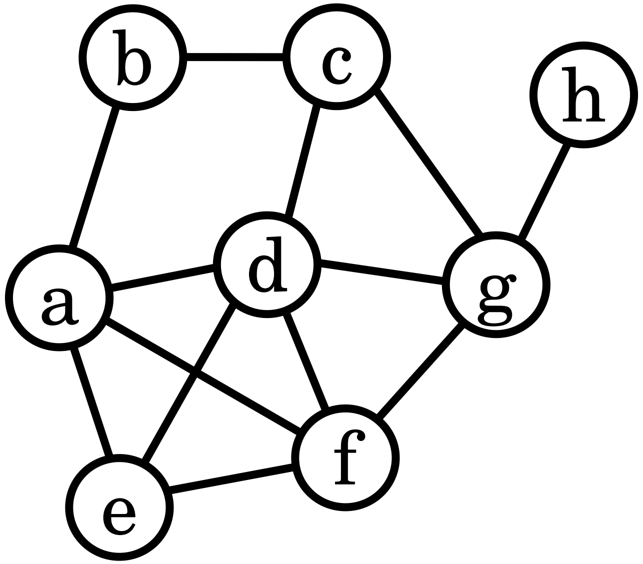


Matches

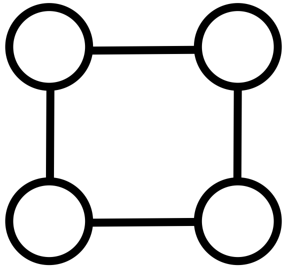


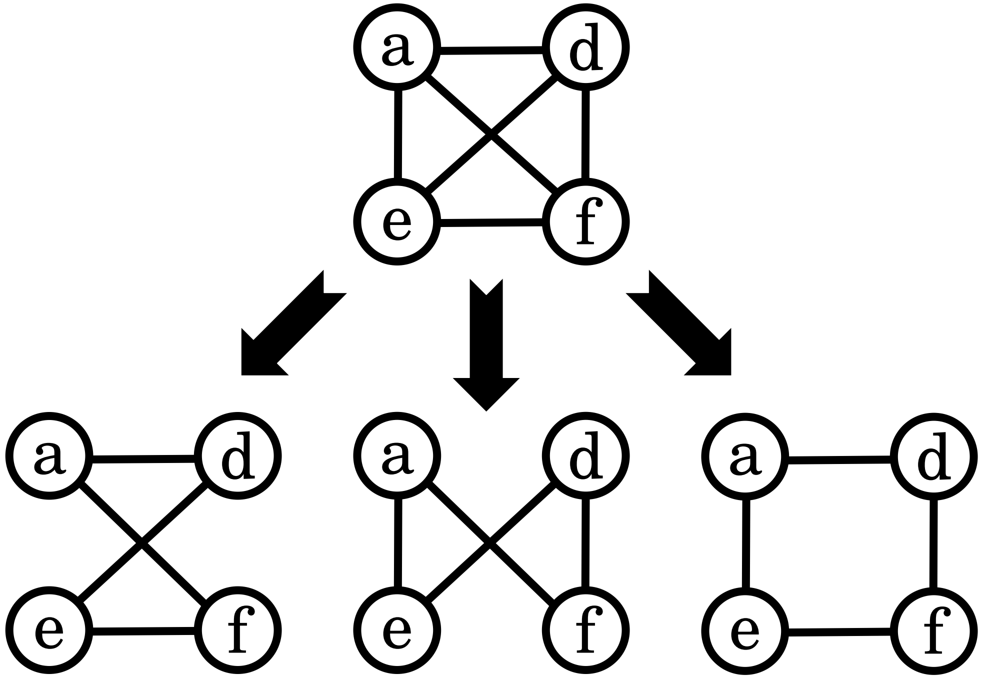
Subgraph Morphing: Theory

Data Graph

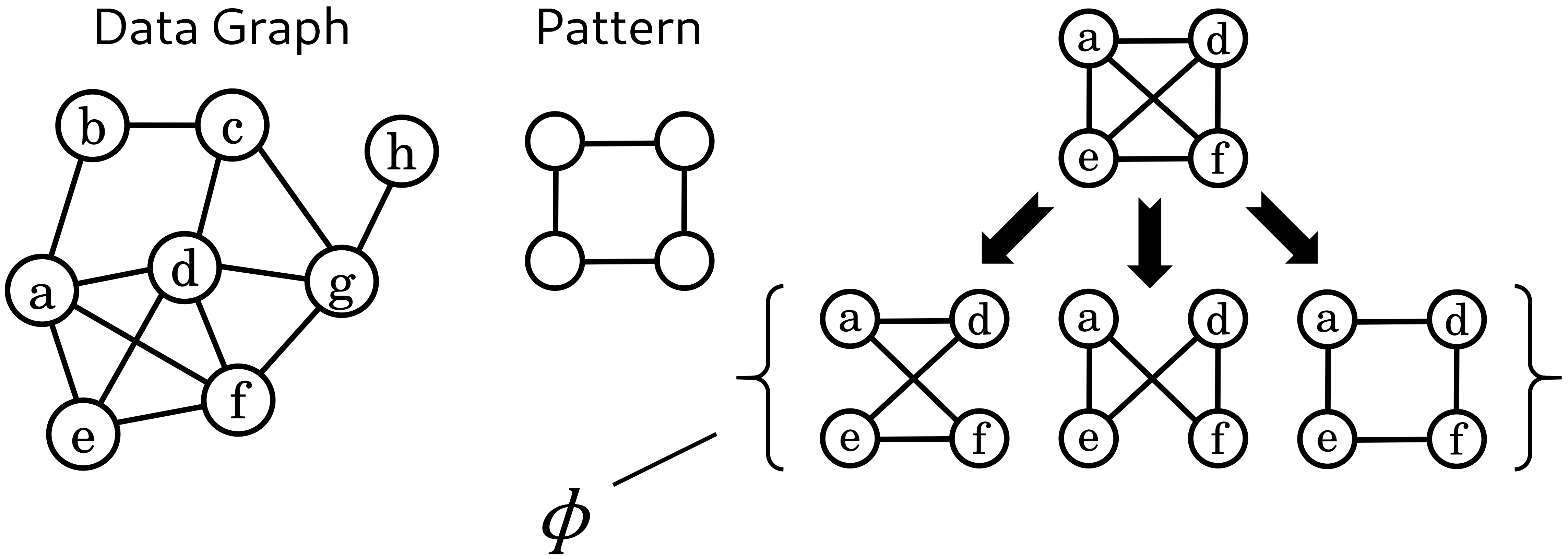


Pattern



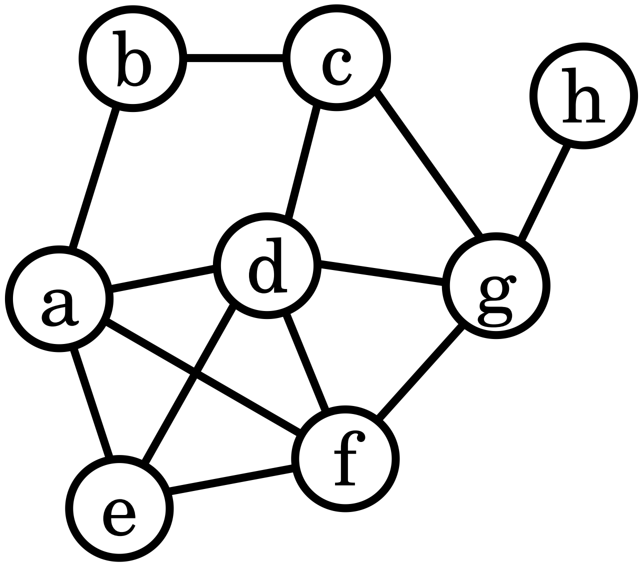


Subgraph Morphing: Theory

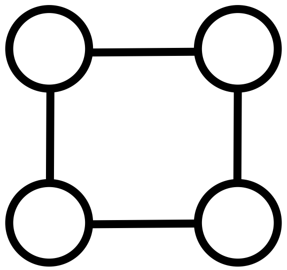


Subgraph Morphing: Theory

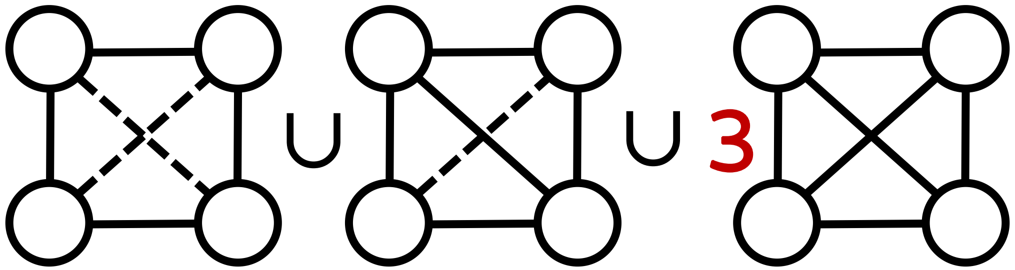
Data Graph



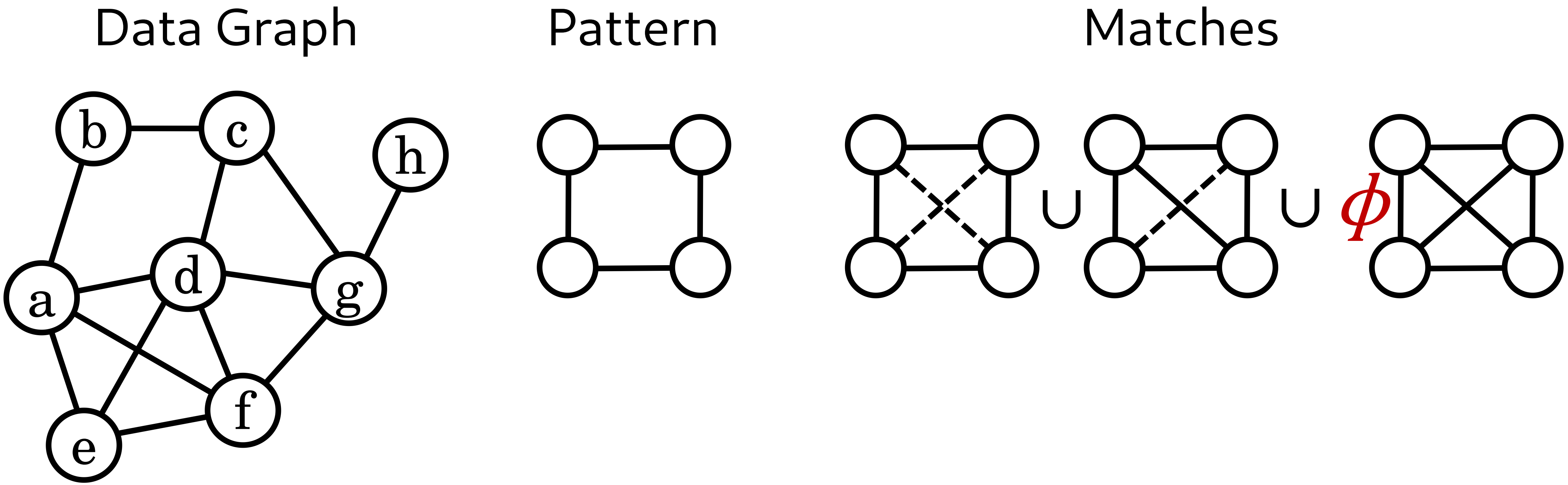
Pattern



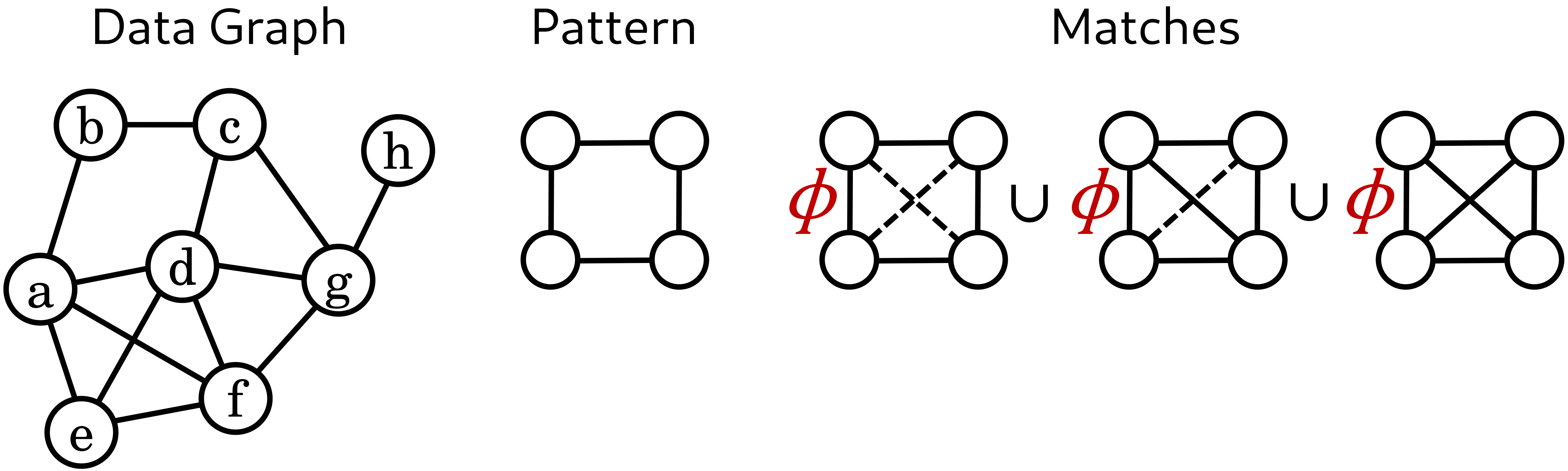
Matches



Subgraph Morphing: Theory

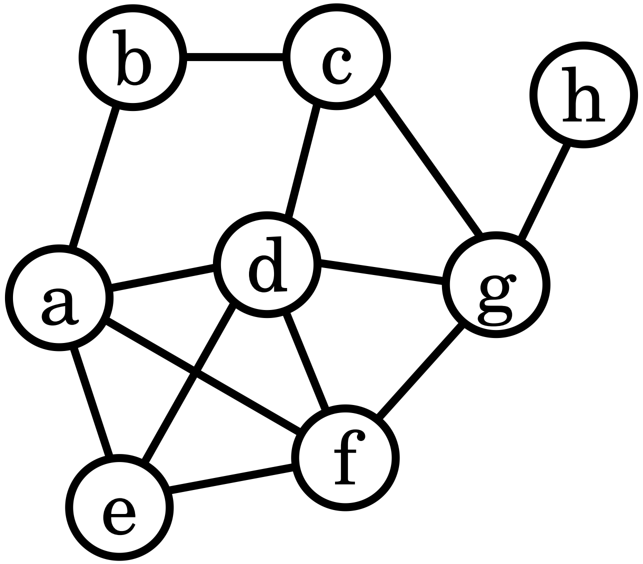


Subgraph Morphing: Theory

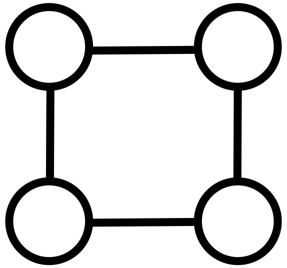


Subgraph Morphing: Theory

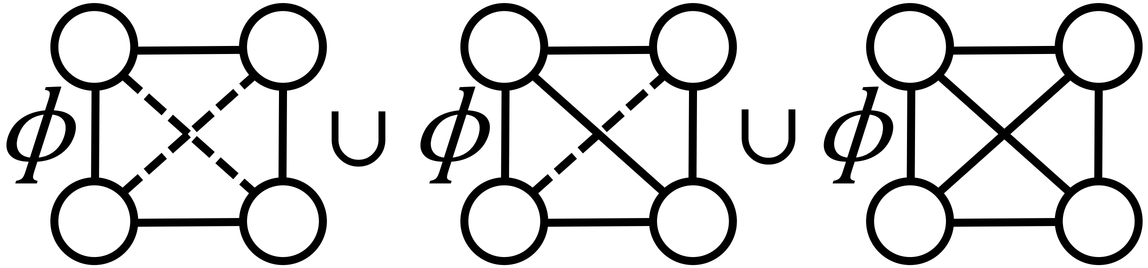
Data Graph



Pattern

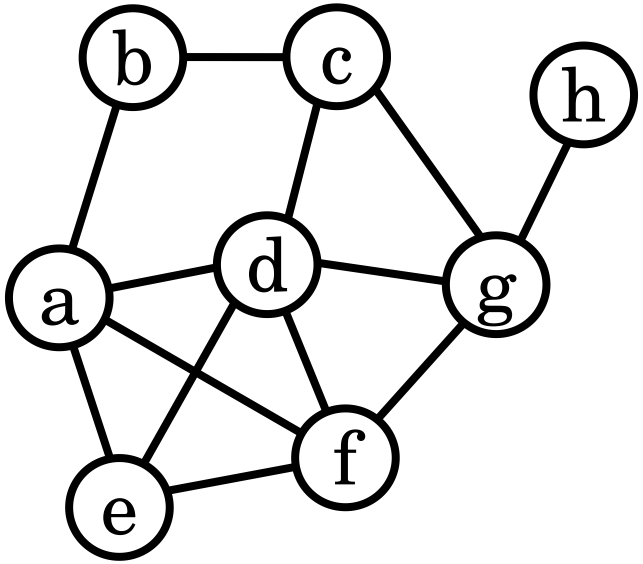


Matches

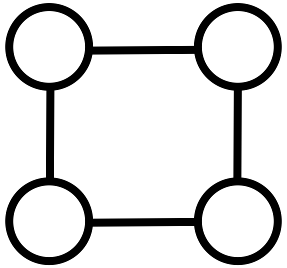


Subgraph Morphing: Theory

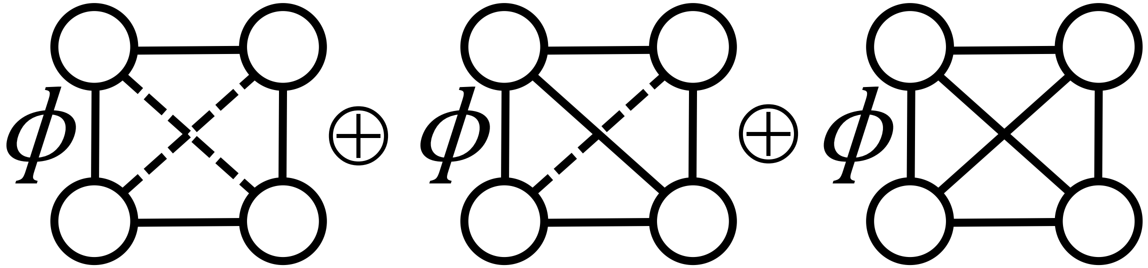
Data Graph



Pattern

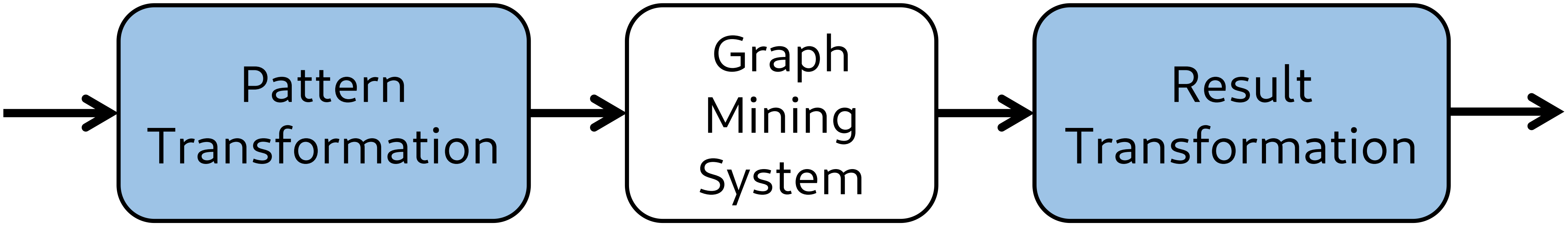


Matches

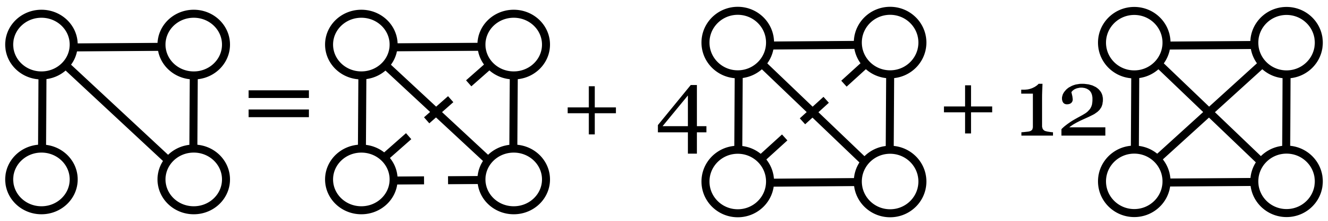


Subgraph Morphing

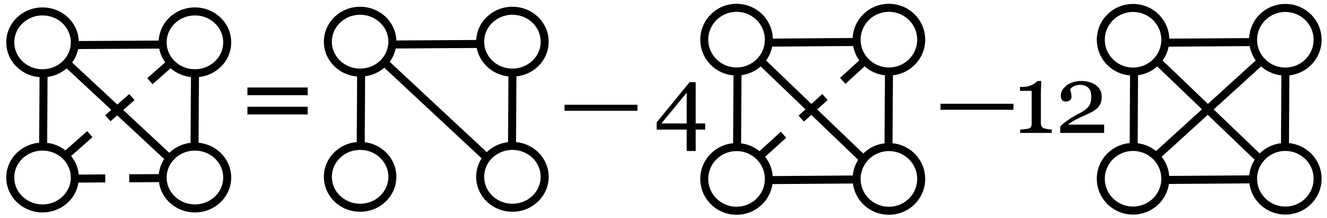
- Replace slow patterns with fast patterns
- Transform results to be consistent with original inputs



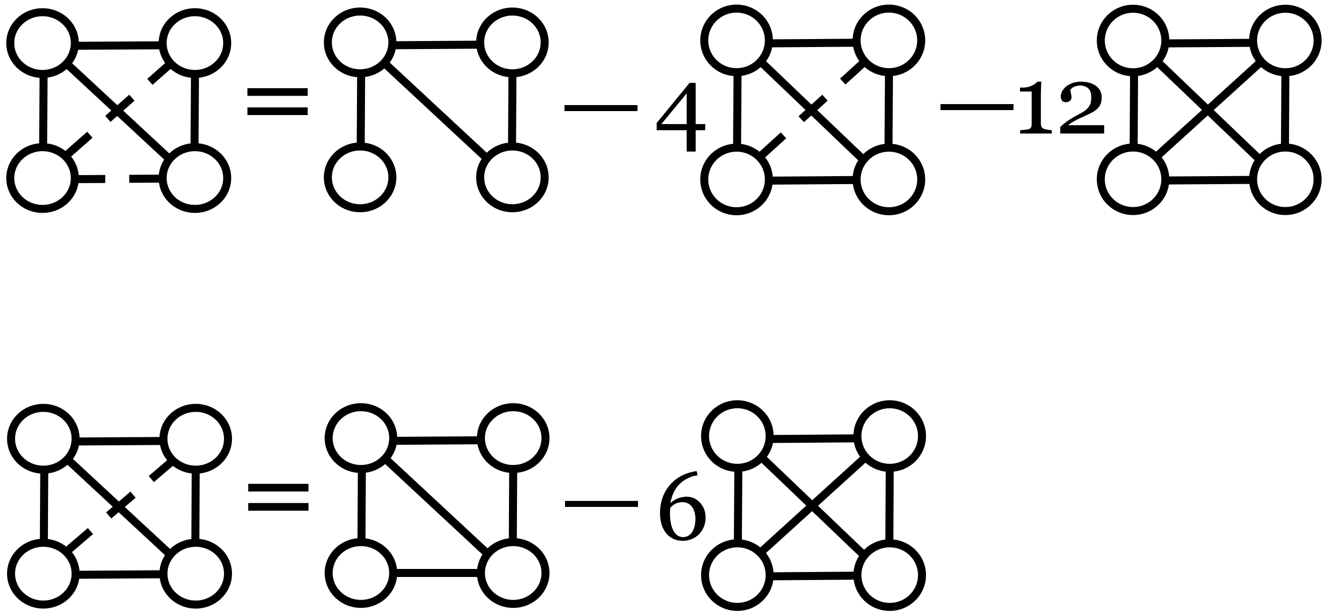
Pattern Transformation



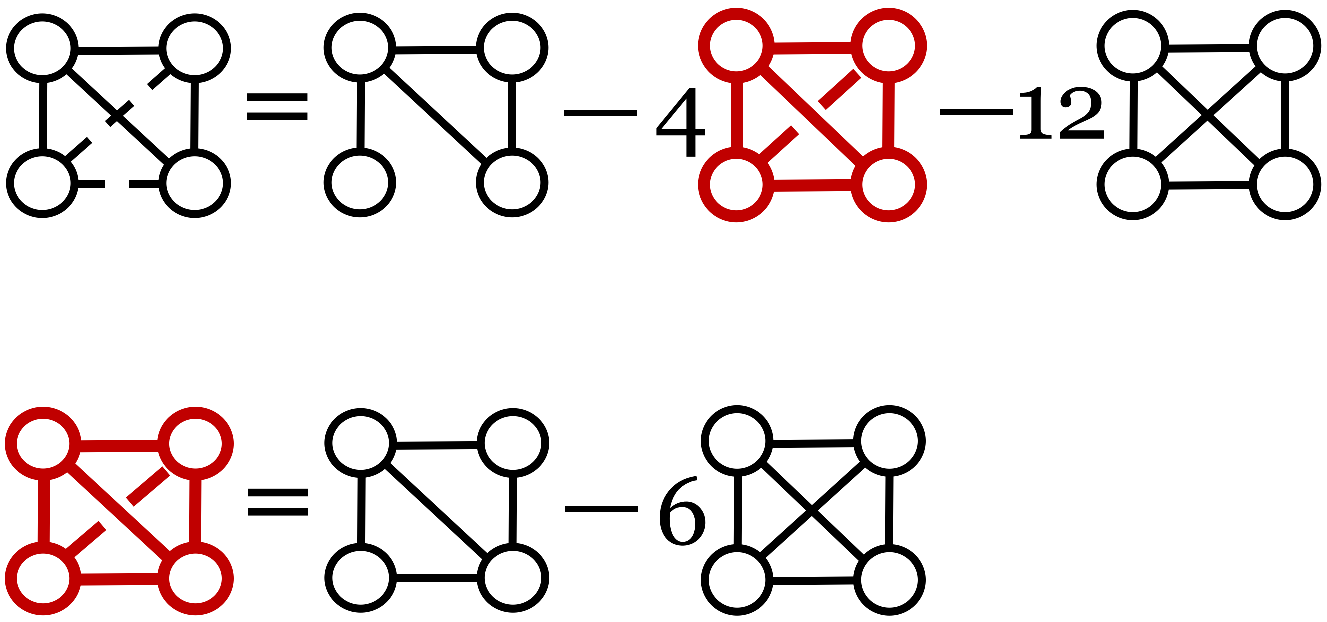
Pattern Transformation



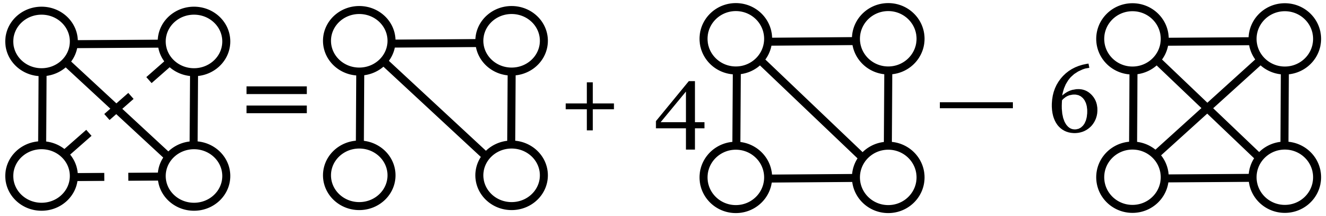
Pattern Transformation

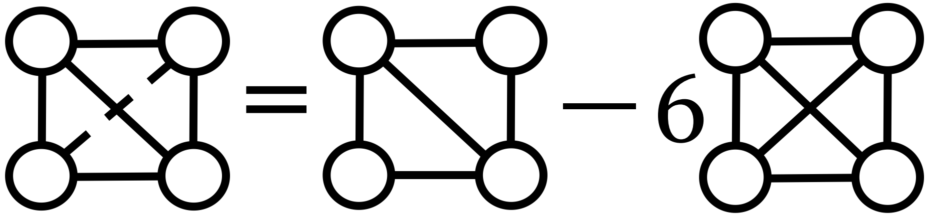


Pattern Transformation

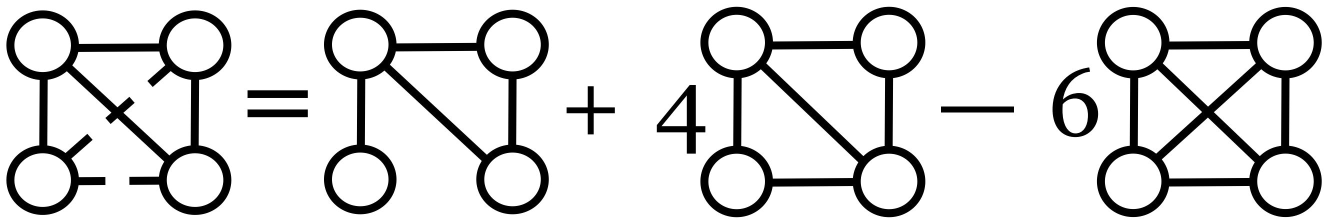


Pattern Transformation

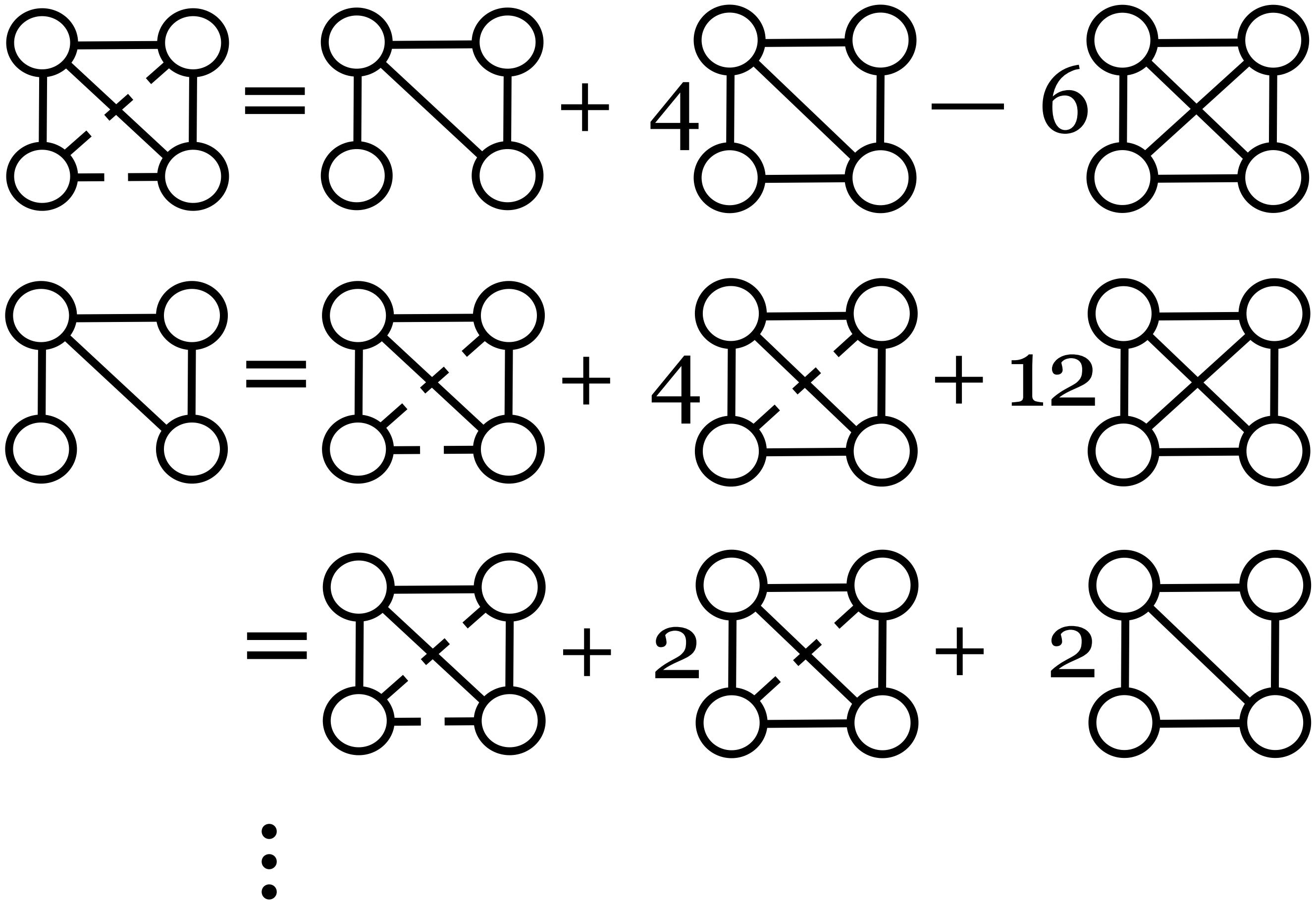




Pattern Transformation



Pattern Transformation



Pattern Transformation

- Explore combinations of different patterns
- Choose an efficient combination

Pattern Transformation

- Explore combinations of different patterns

- Choose an **efficient** combination

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

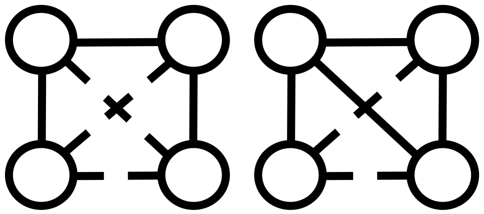
- Tap into the underyling system’s cost-based optimizer
- Account for UDF costs (i.e., per-match overhead)

Pattern Transformation

- Explore combinations of different patterns
- Choose an efficient combination

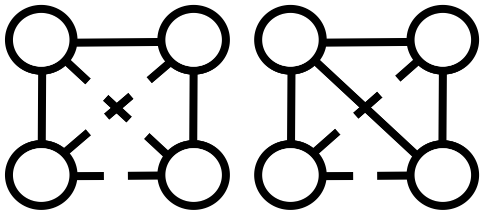
Pattern Transformation

Input Patterns

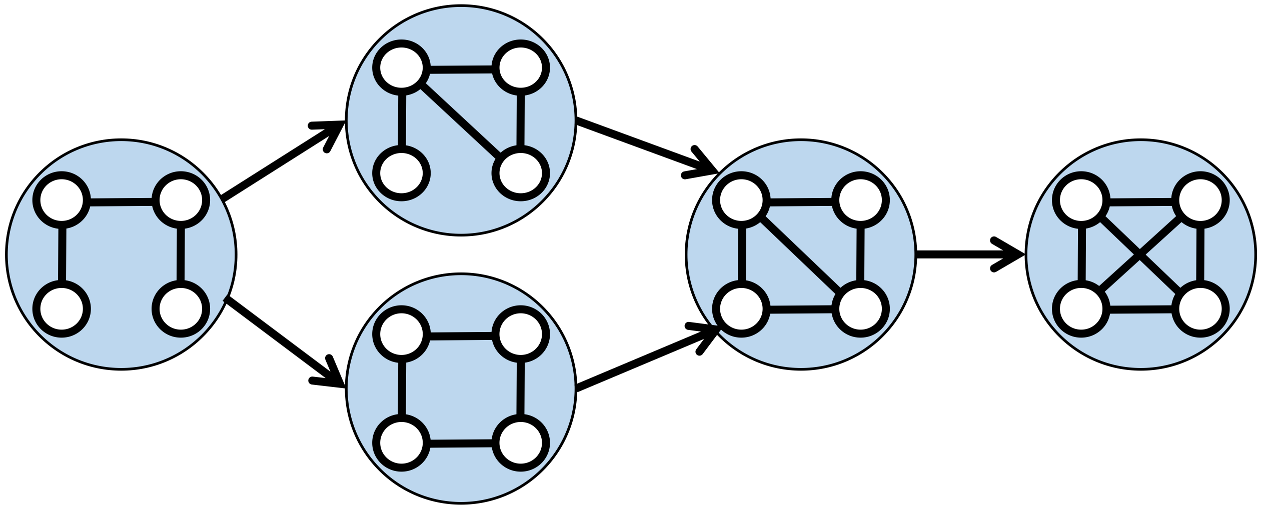


Pattern Transformation

Input Patterns

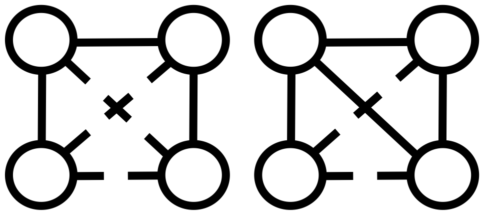


S-DAG

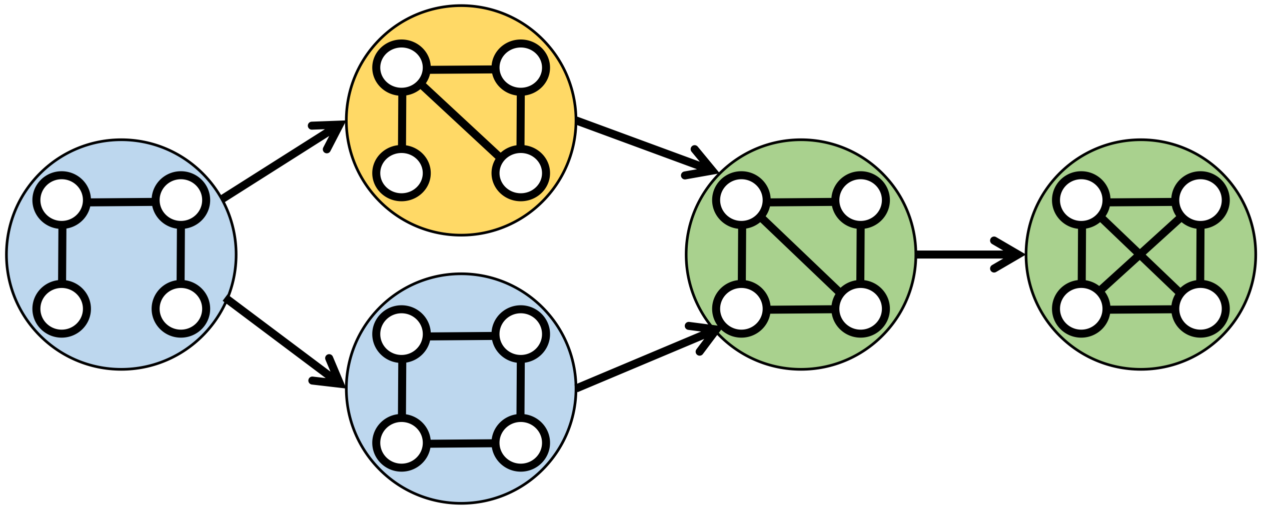


Pattern Transformation

Input Patterns

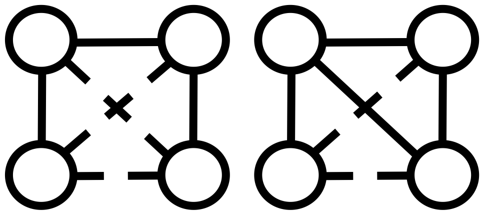


S-DAG

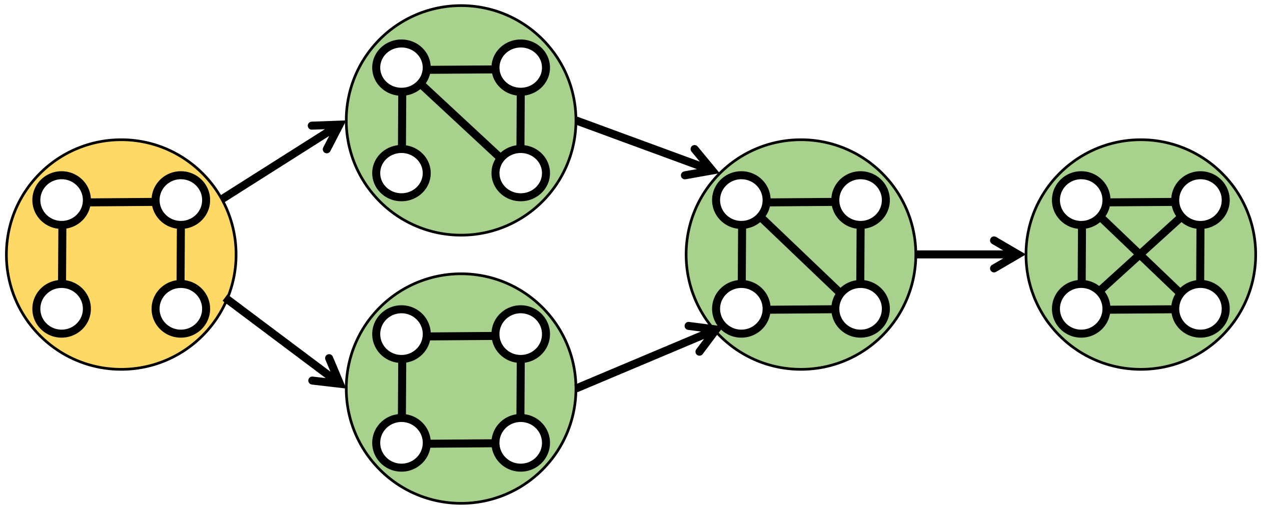


Pattern Transformation

Input Patterns

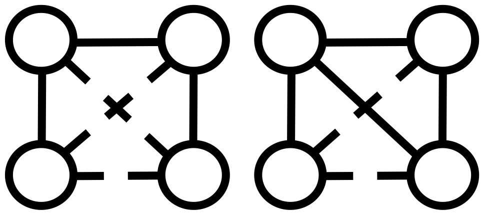


S-DAG

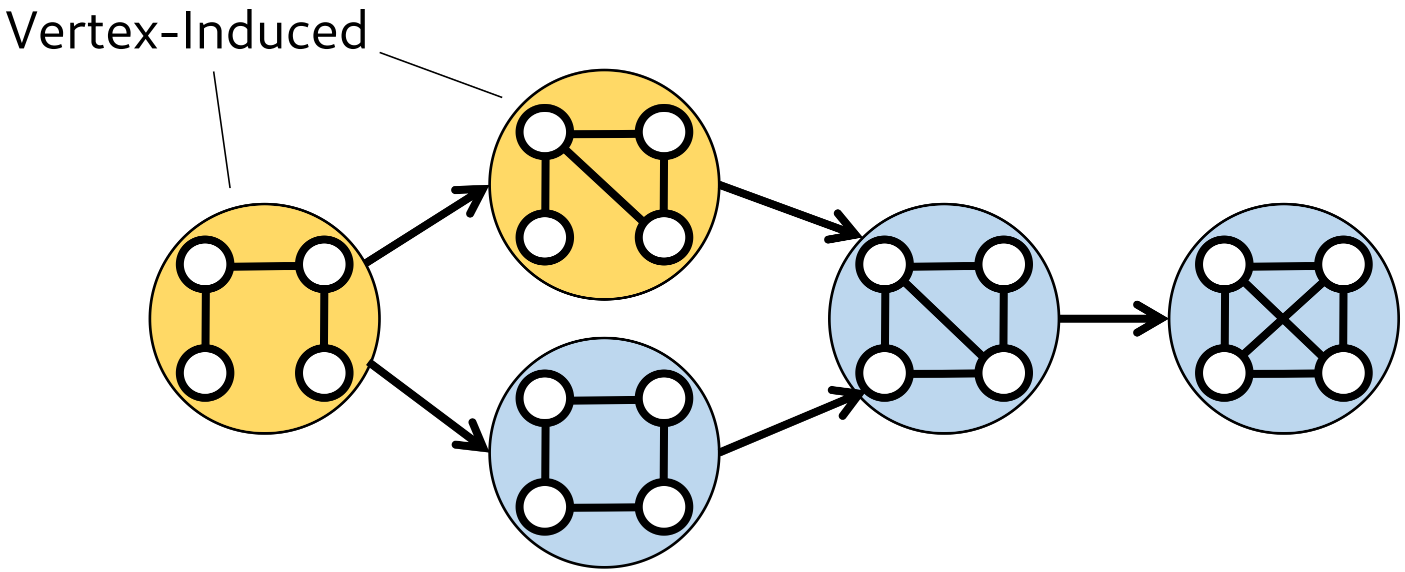


Pattern Transformation

Input Patterns

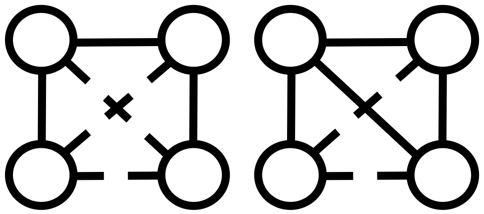


S-DAG

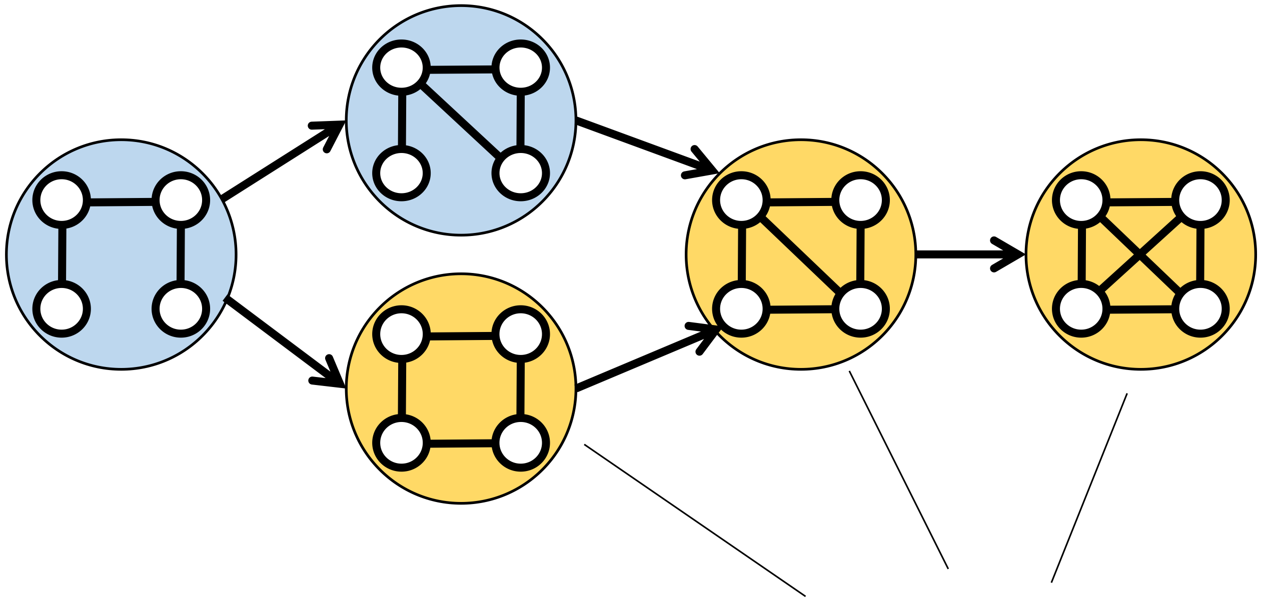


Pattern Transformation

Input Patterns



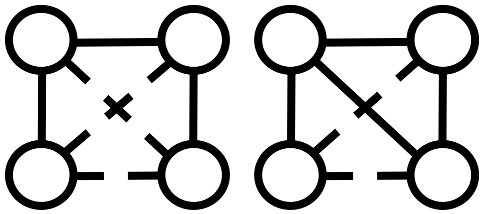
S-DAG



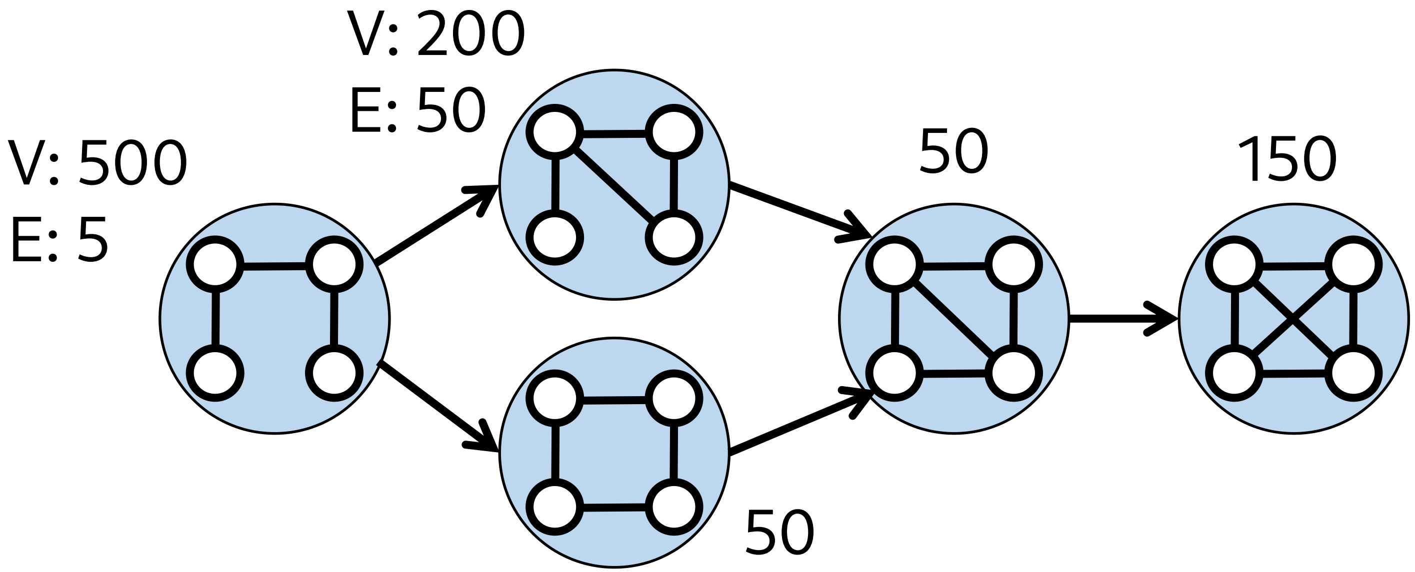
Whichever variant has lower cost

Pattern Transformation

Input Patterns

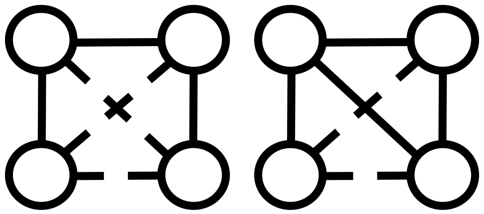


S-DAG



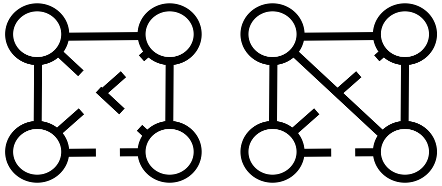
Pattern Transformation

Input Patterns

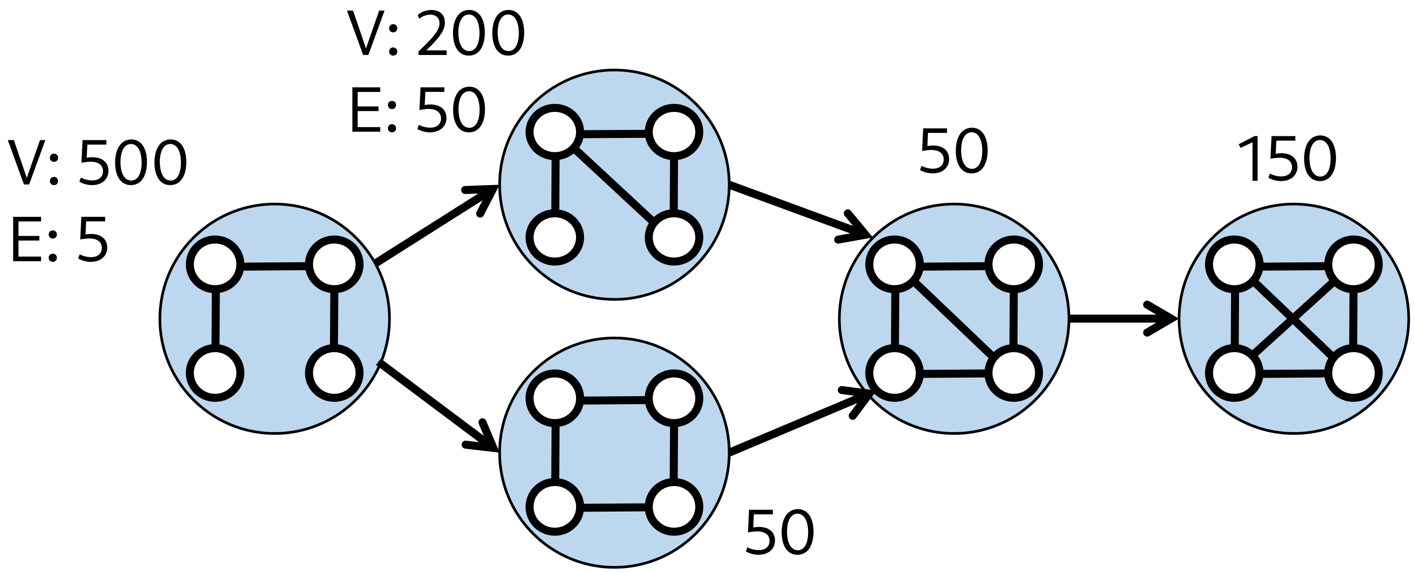


Alternative Patterns

Cost: 700

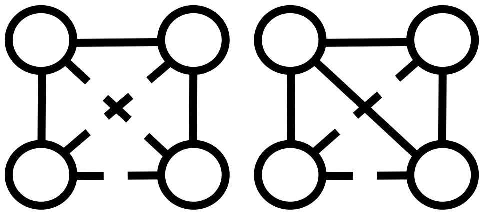


S-DAG



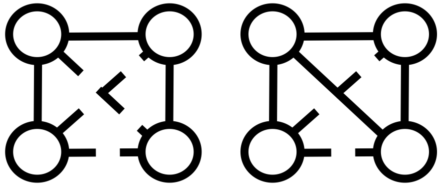
Pattern Transformation

Input Patterns

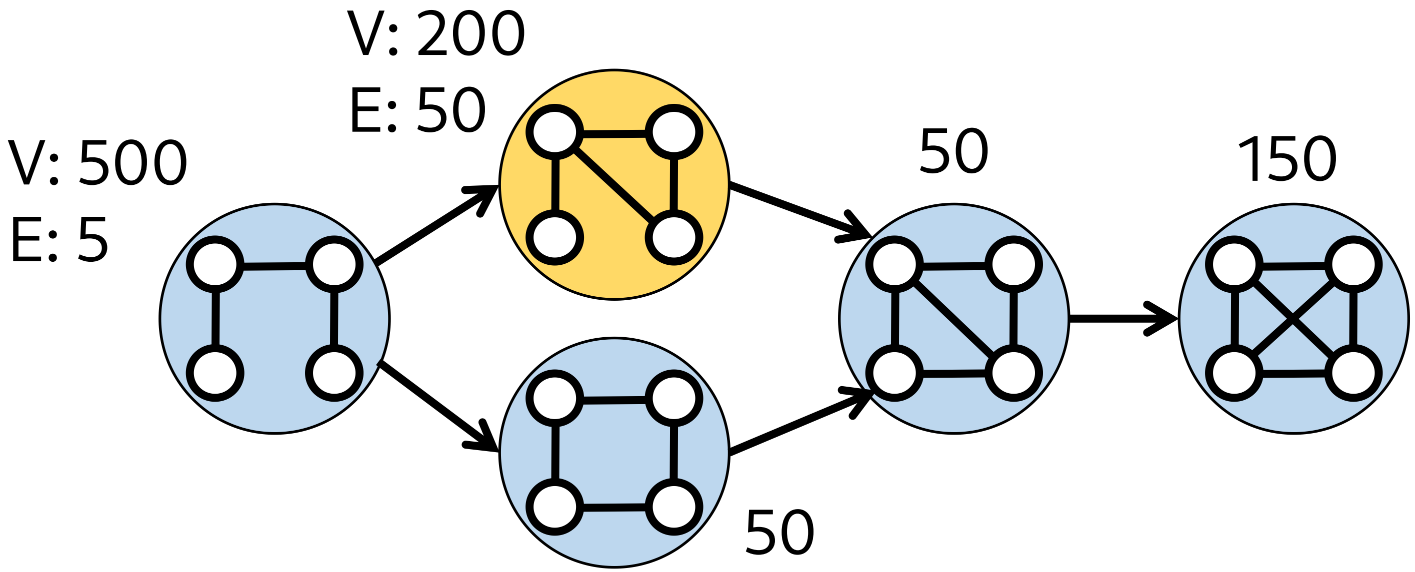


Alternative Patterns

Cost: 700

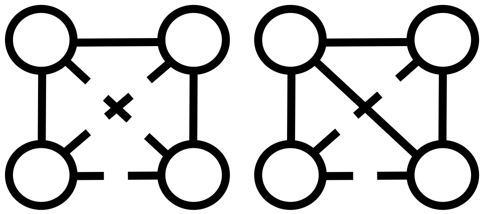


S-DAG



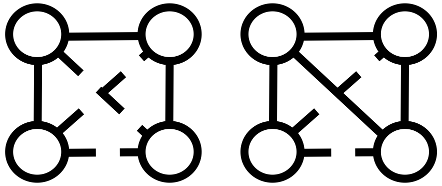
Pattern Transformation

Input Patterns

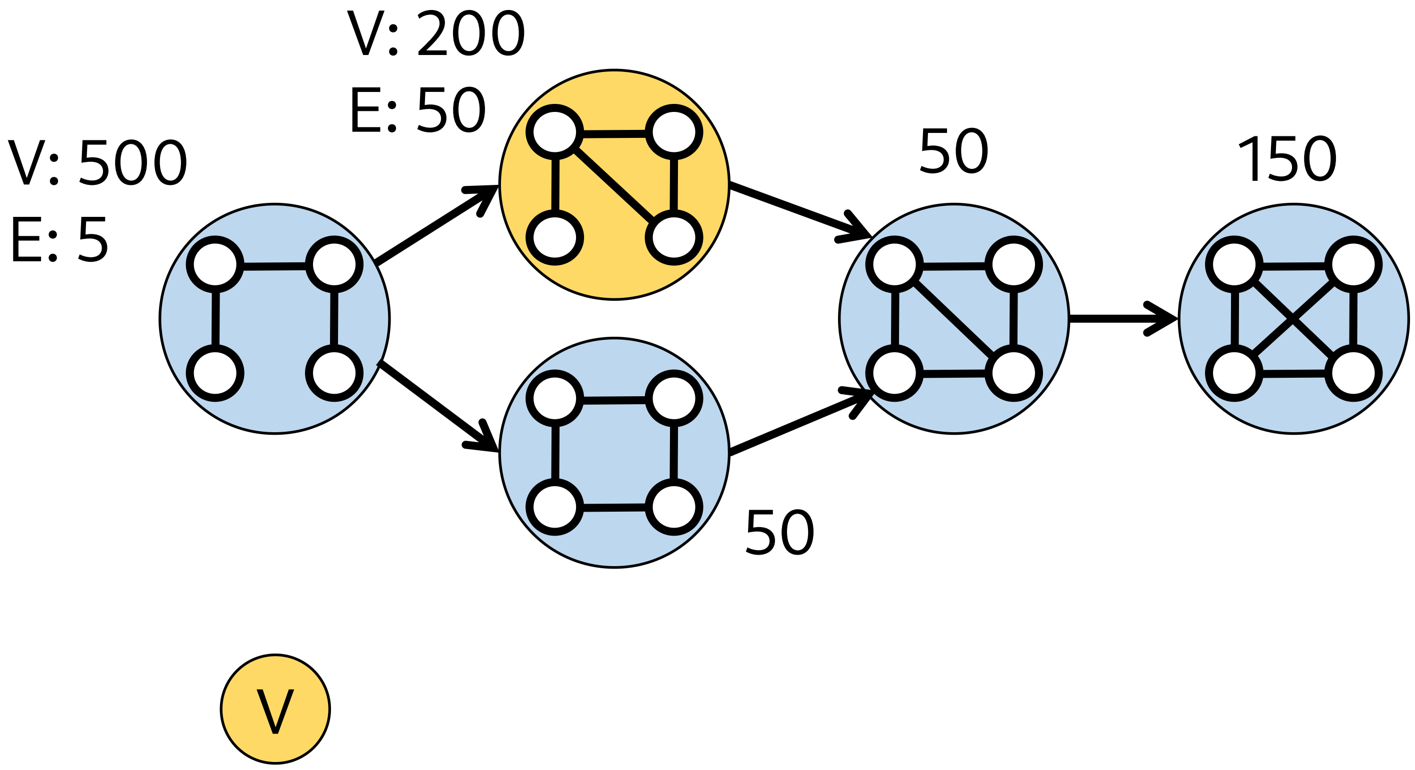


Alternative Patterns

Cost: 700

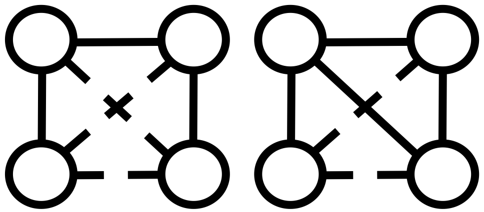


S-DAG



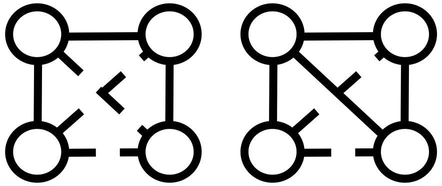
Pattern Transformation

Input Patterns

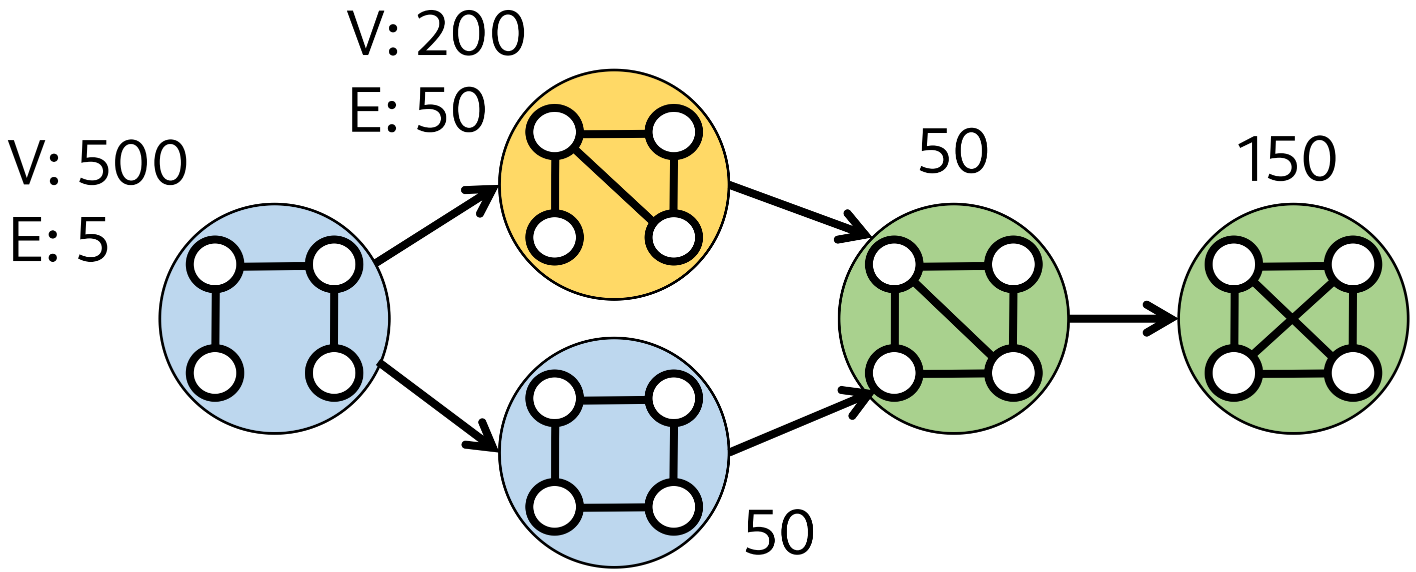


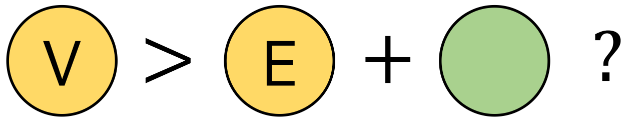
Alternative Patterns

Cost: 700



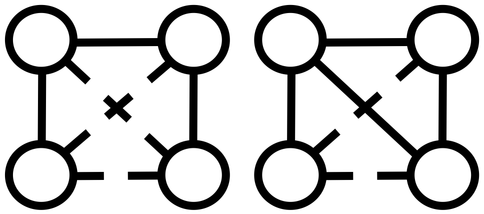
S-DAG





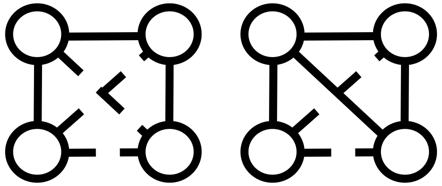
Pattern Transformation

Input Patterns

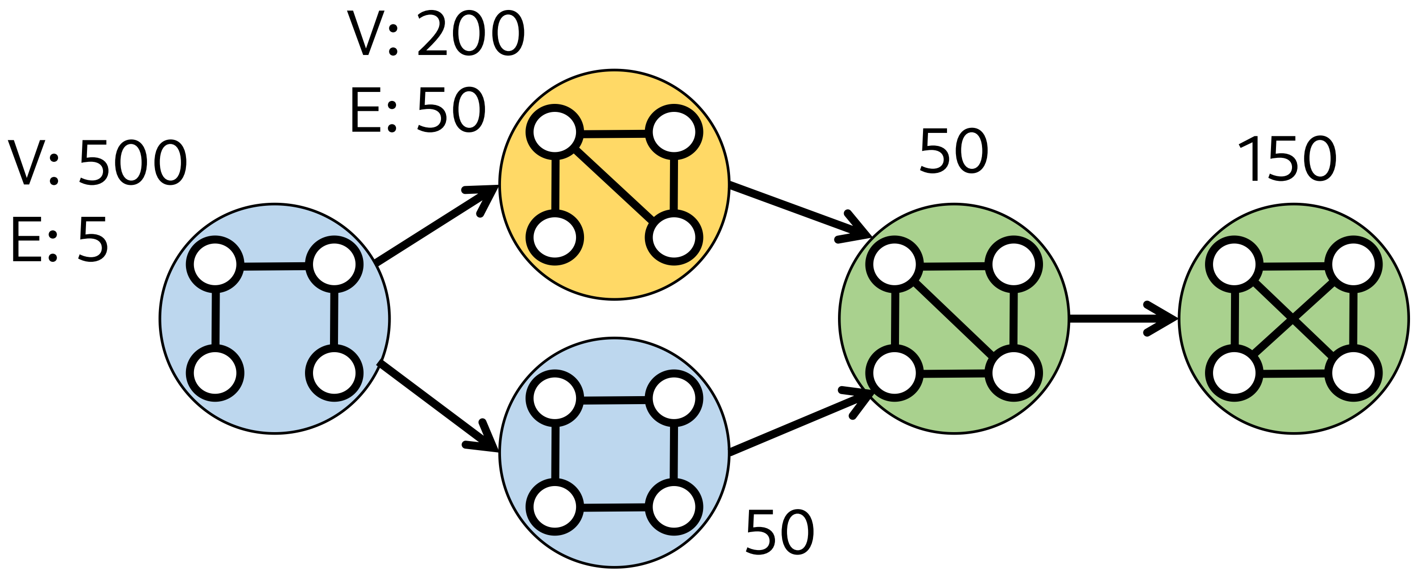


Alternative Patterns

Cost: 700



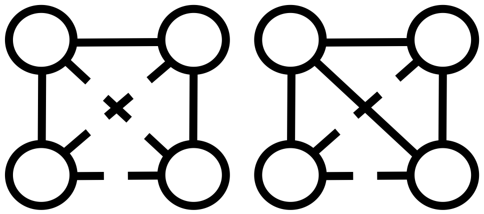
S-DAG



200 > 50 + 50 + 150 ?

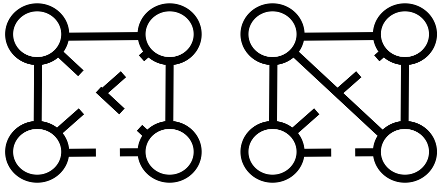
Pattern Transformation

Input Patterns

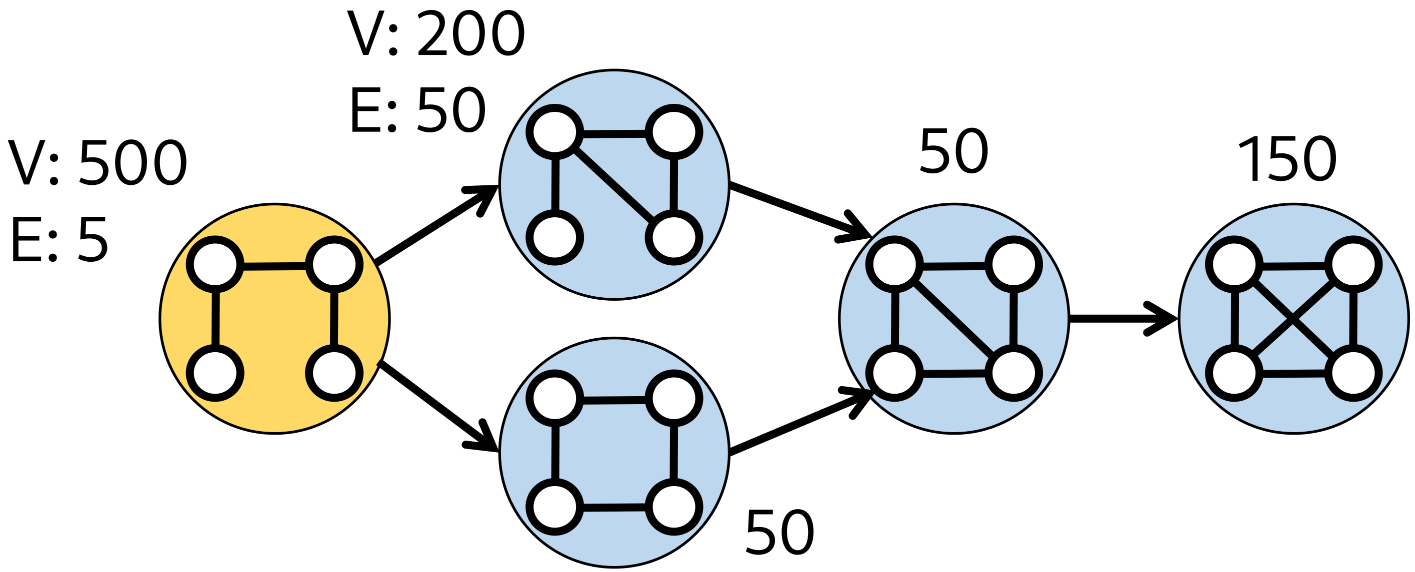


Alternative Patterns

Cost: 700

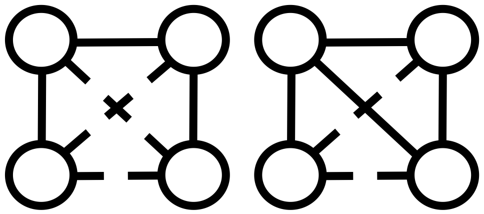


S-DAG



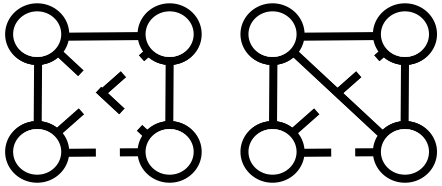
Pattern Transformation

Input Patterns

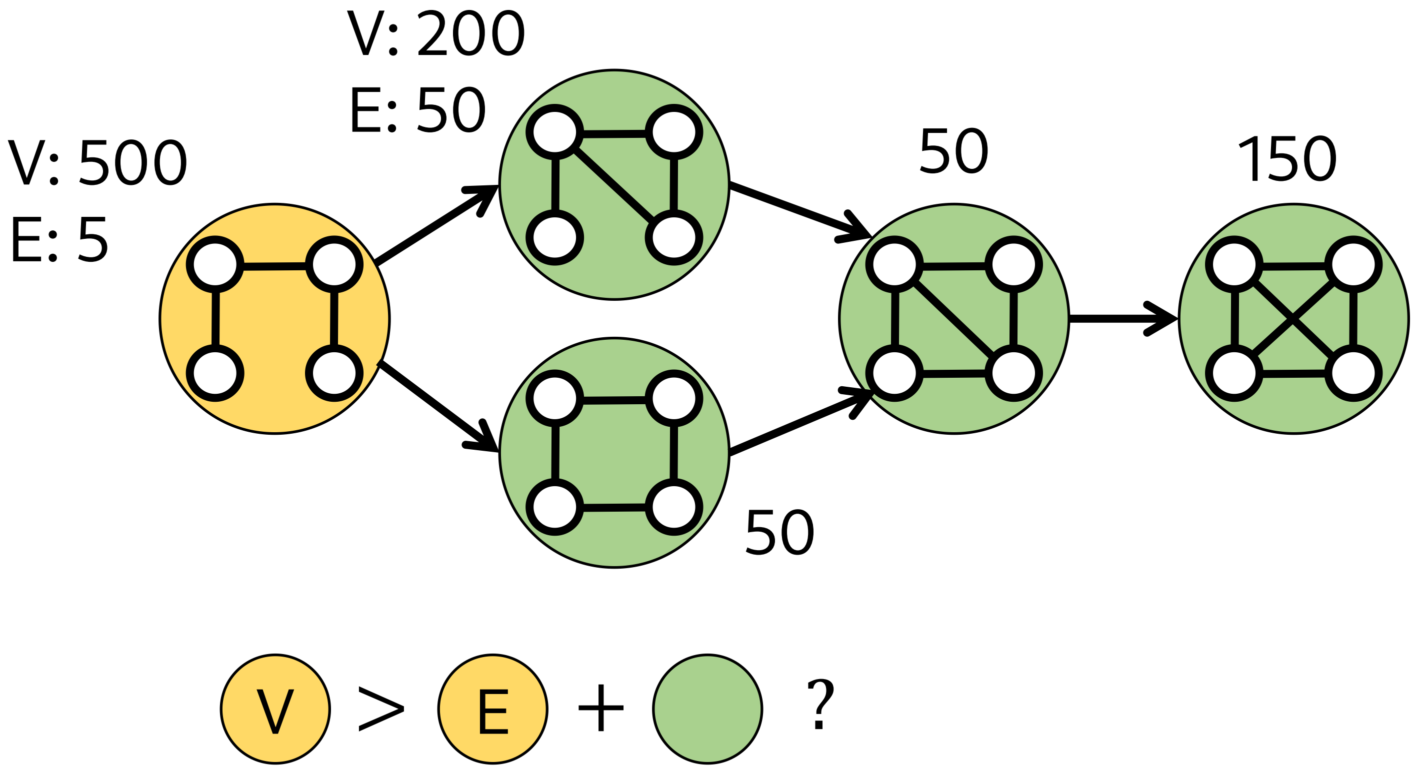


Alternative Patterns

Cost: 700

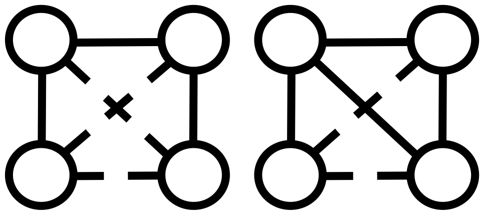


S-DAG



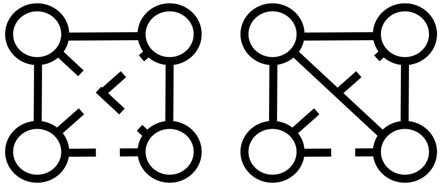
Pattern Transformation

Input Patterns

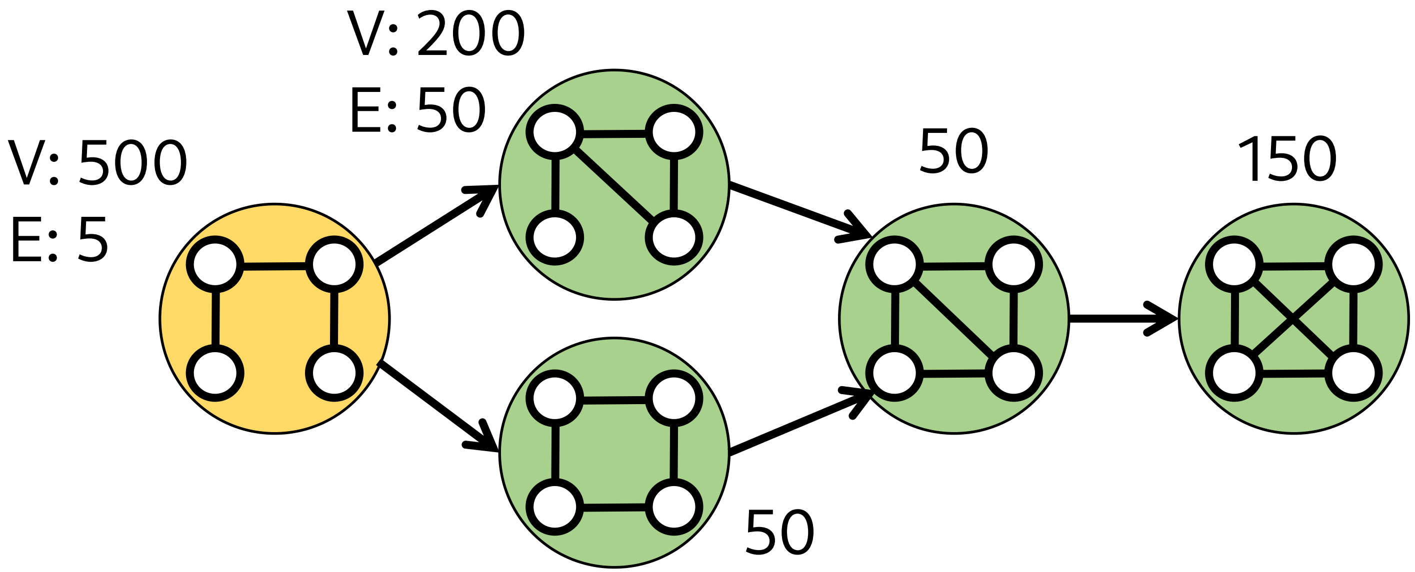


Alternative Patterns

Cost: 700



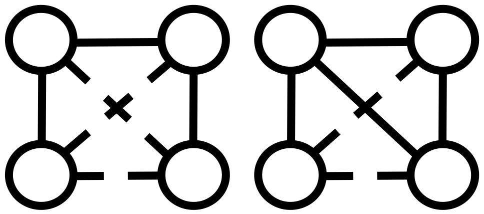
S-DAG



500 > 5 + 450 ?

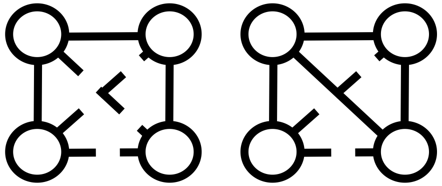
Pattern Transformation

Input Patterns

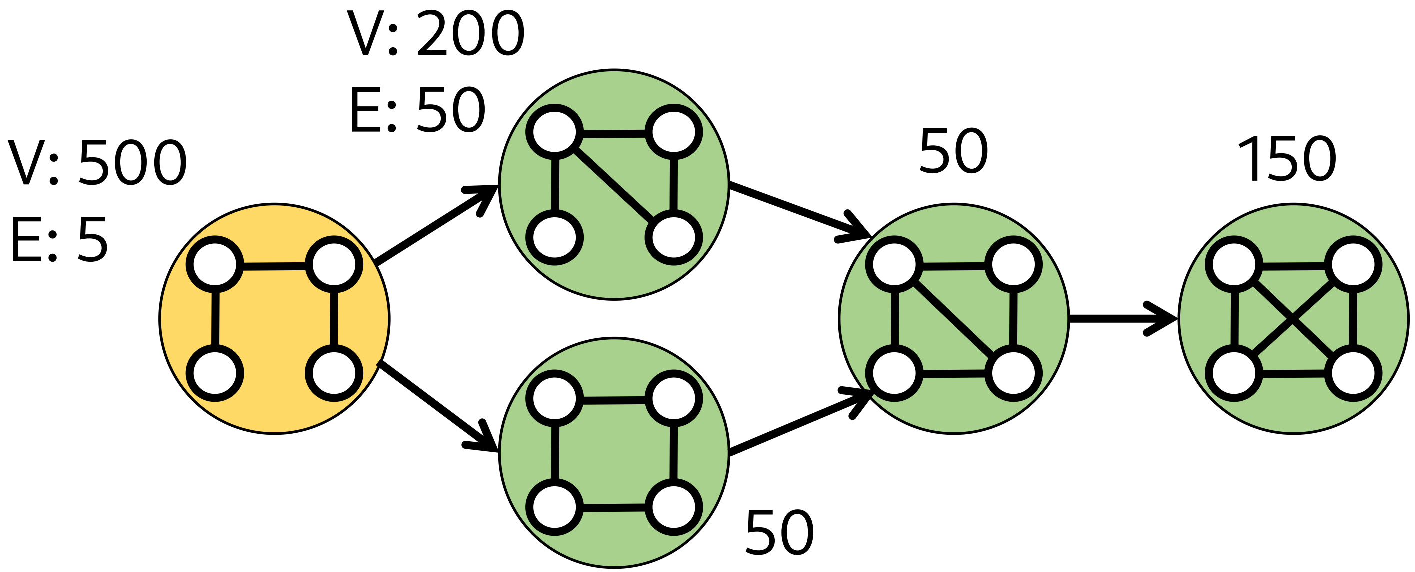


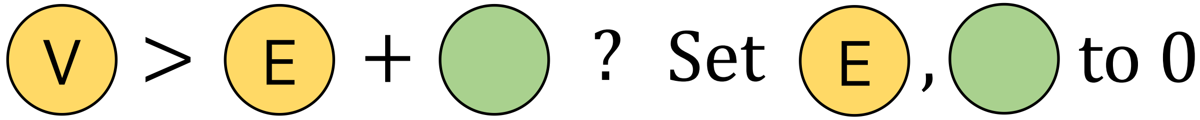
Alternative Patterns

Cost: 700



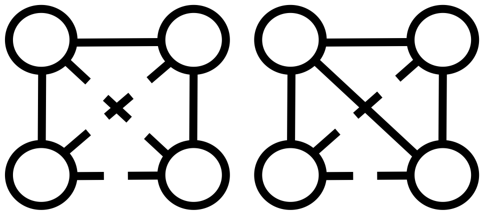
S-DAG





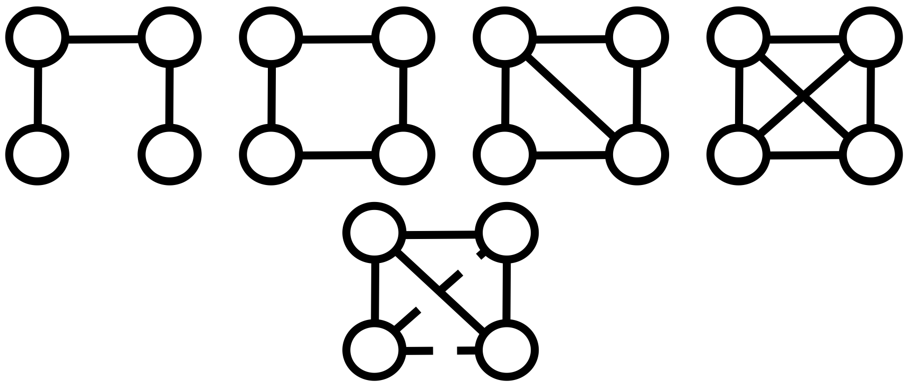
Pattern Transformation

Input Patterns

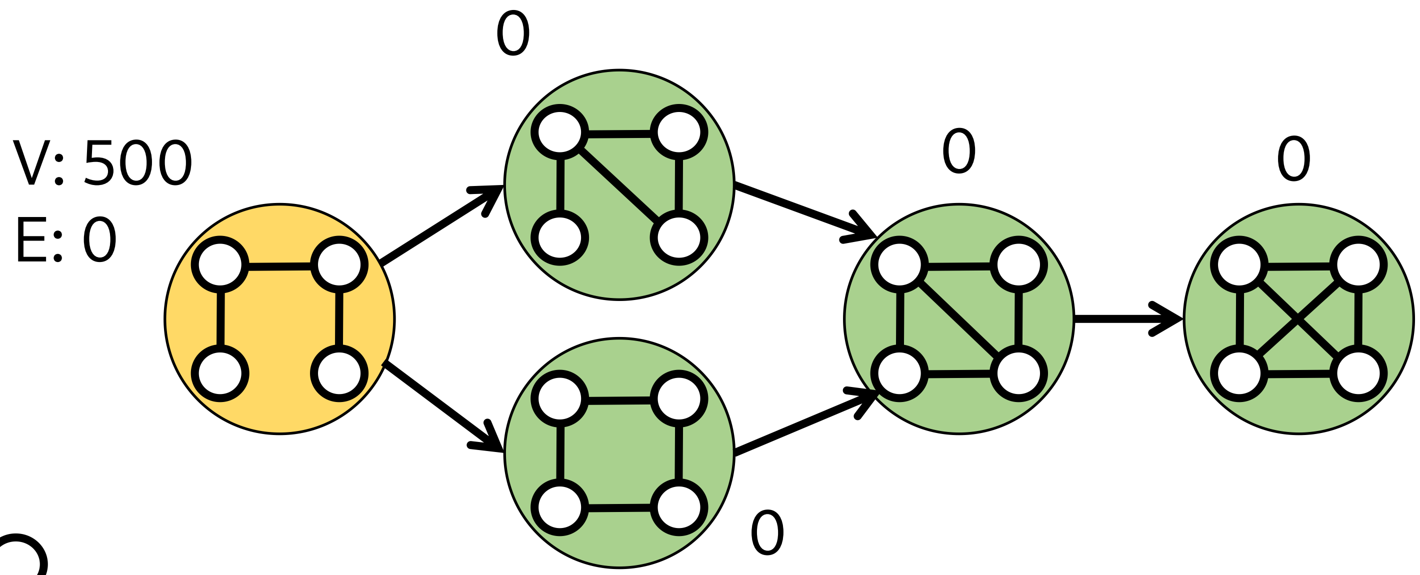


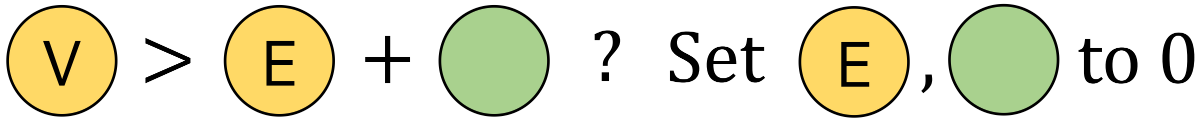
Alternative Patterns

Cost: 455



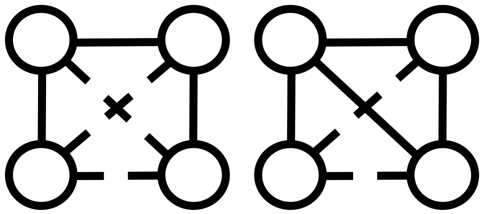
S-DAG





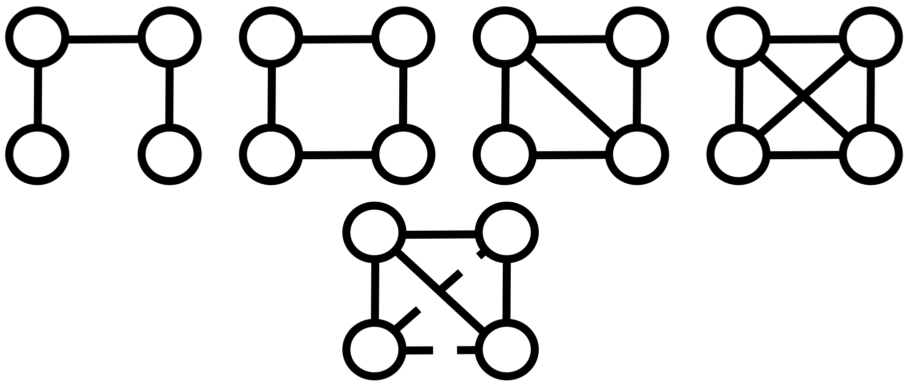
Pattern Transformation

Input Patterns

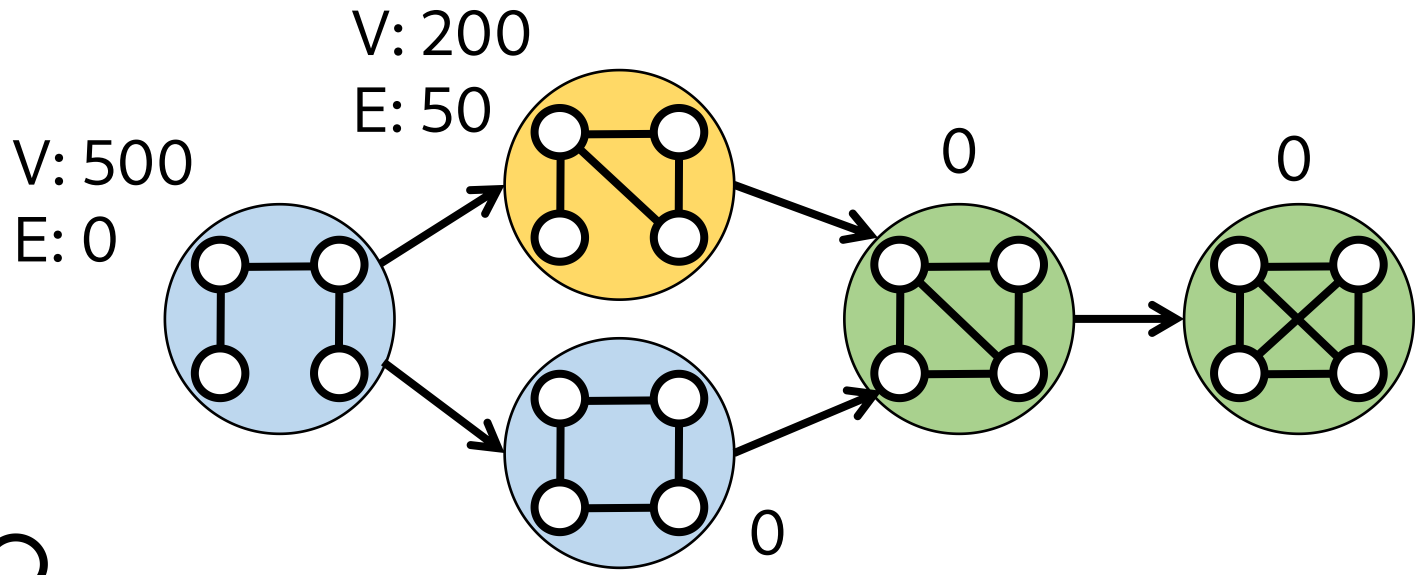


Alternative Patterns

Cost: 455

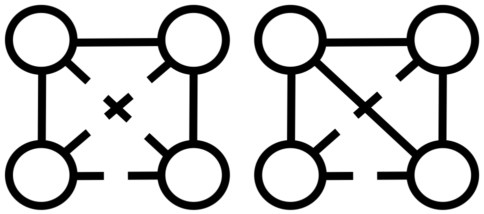


S-DAG



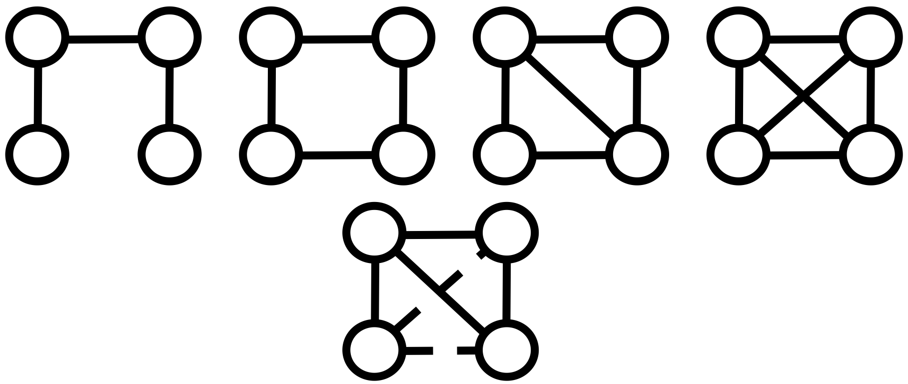
Pattern Transformation

Input Patterns

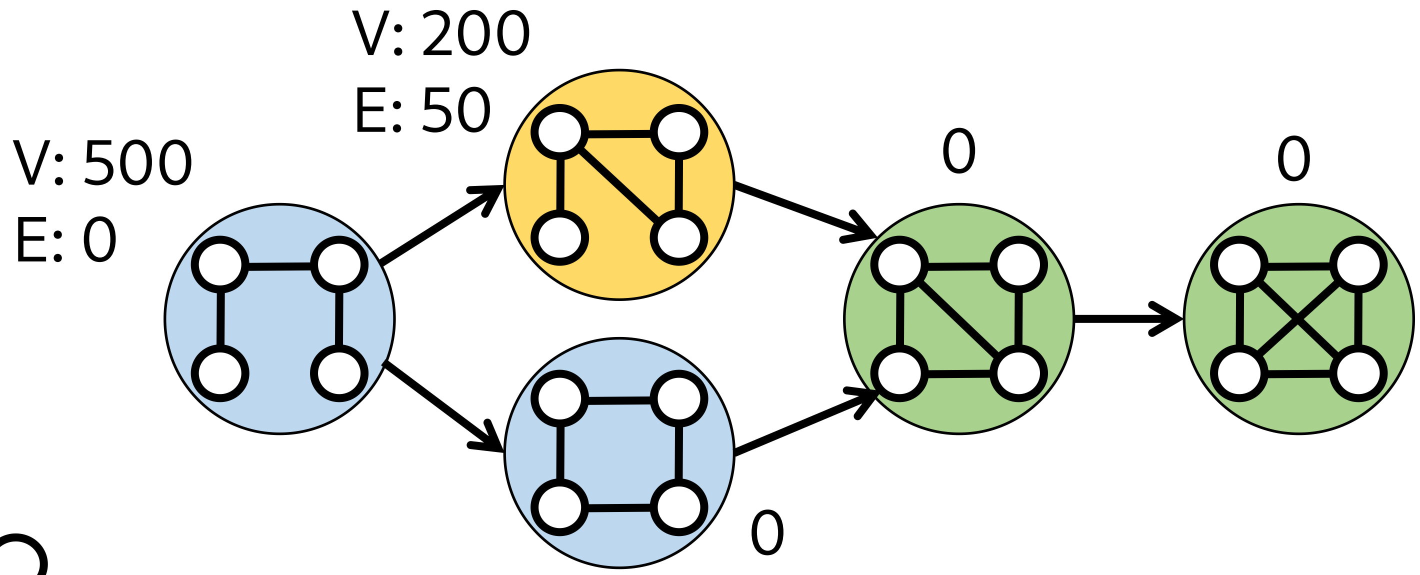


Alternative Patterns

Cost: 455



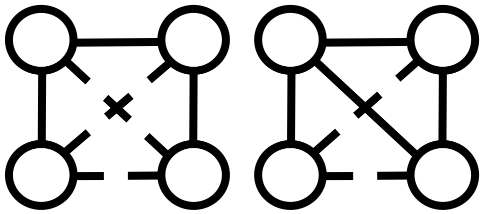
S-DAG



V > E ?

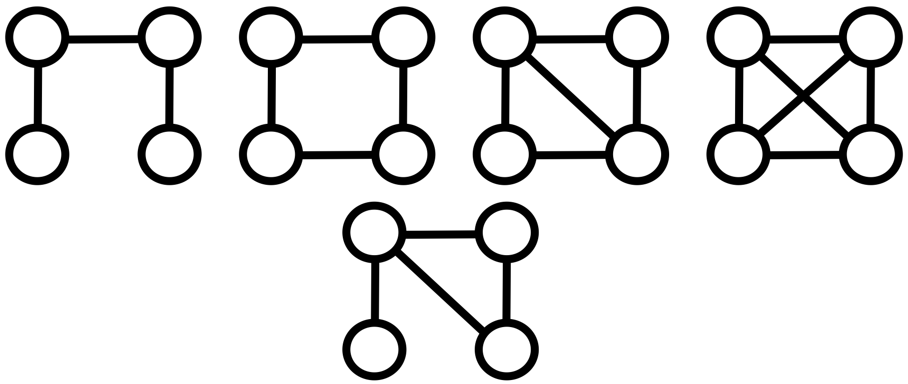
Pattern Transformation

Input Patterns

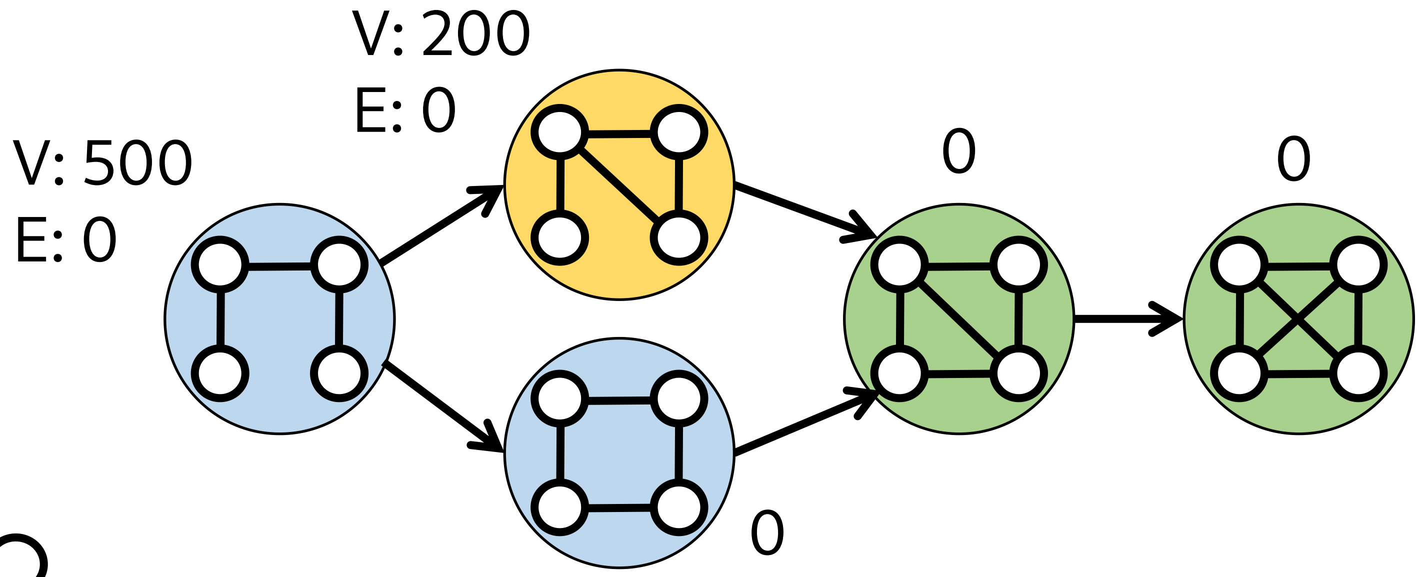


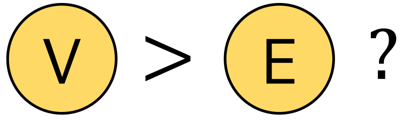
Alternative Patterns

Cost: 305



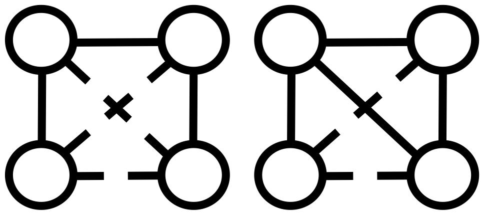
S-DAG



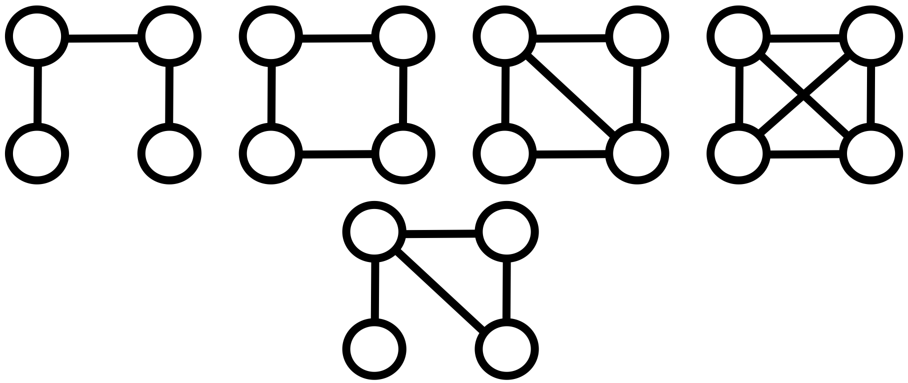


Pattern Transformation

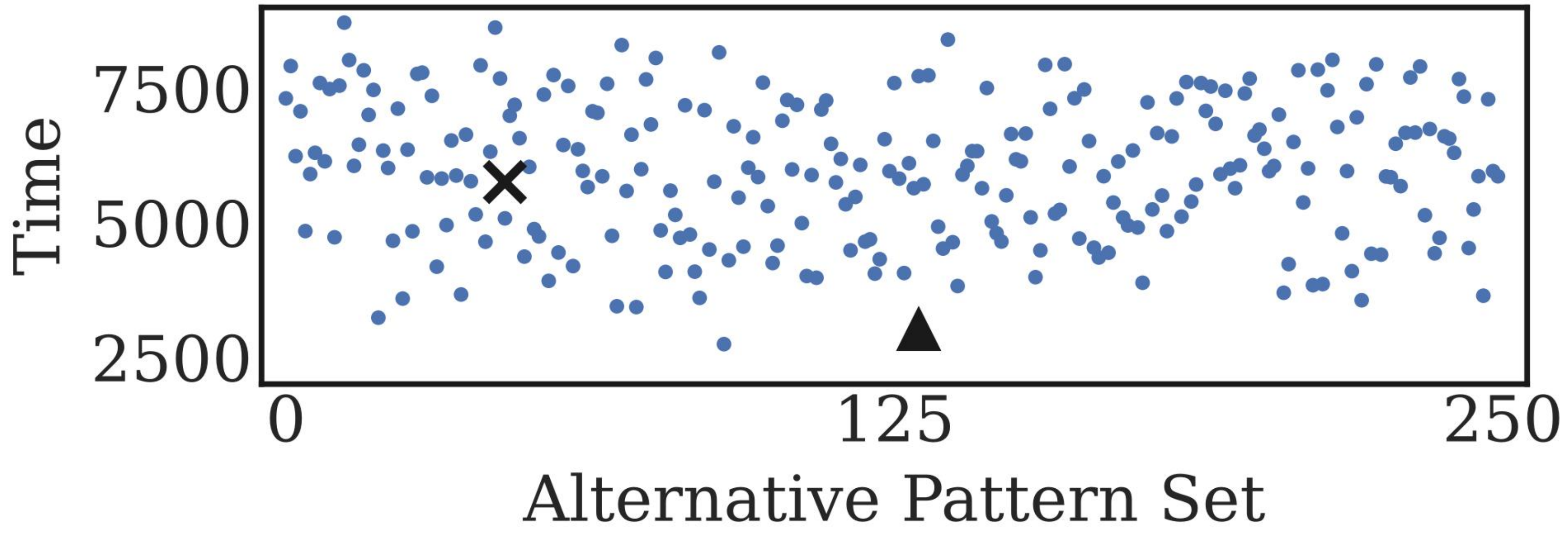
Input Patterns



Alternative Patterns



Pattern Transformation



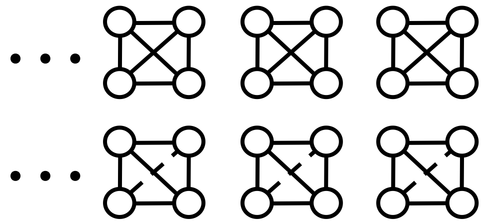
Result Transformation

Result Transformation

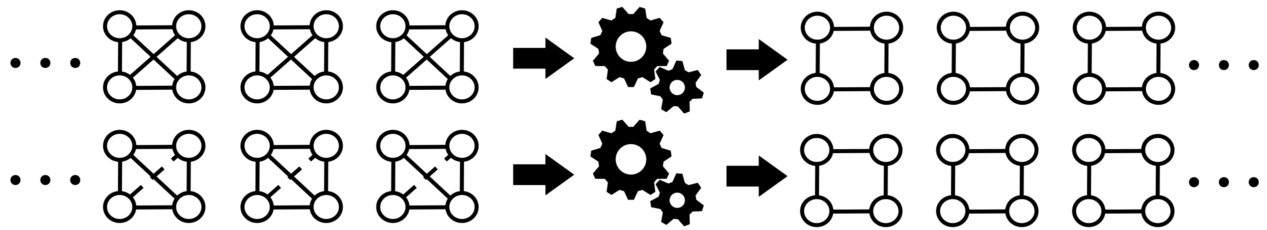
- On-the-Fly

Result Transformation

- On-the-Fly
  - Match alternative patterns

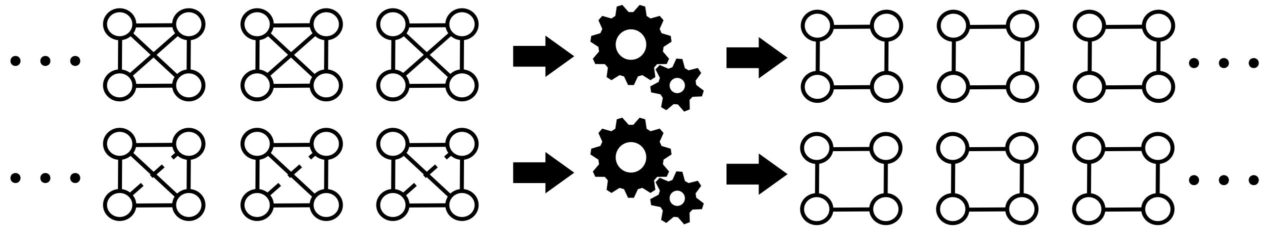


Result Transformation



- On-the-Fly
  - Match alternative patterns
  - Permute each match according to ϕ before applying UDF

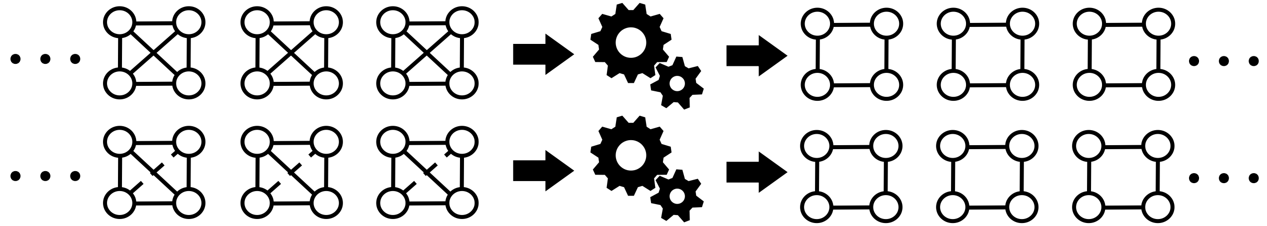
Result Transformation



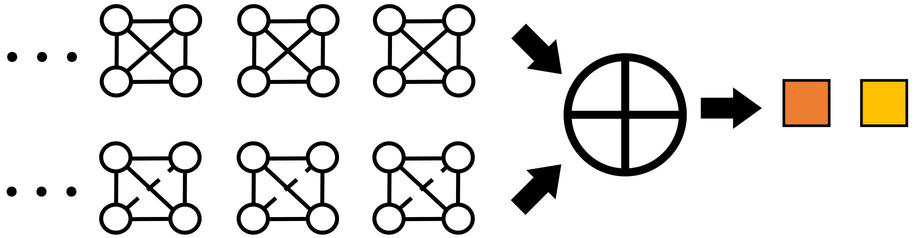
- On-the-Fly
  - Match alternative patterns
  - Permute each match according to ϕ before applying UDF
- Post-Matching

Result Transformation

- On-the-Fly
  - Match alternative patterns
  - Permute each match according to ϕ before applying UDF

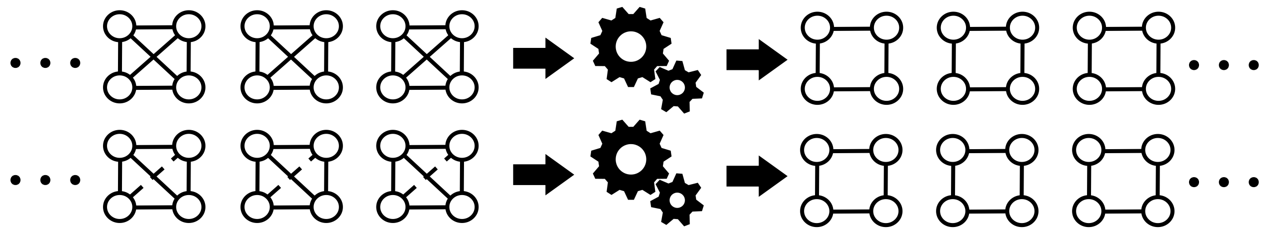


- Post-Matching
  - Match alternative patterns and apply UDF normally

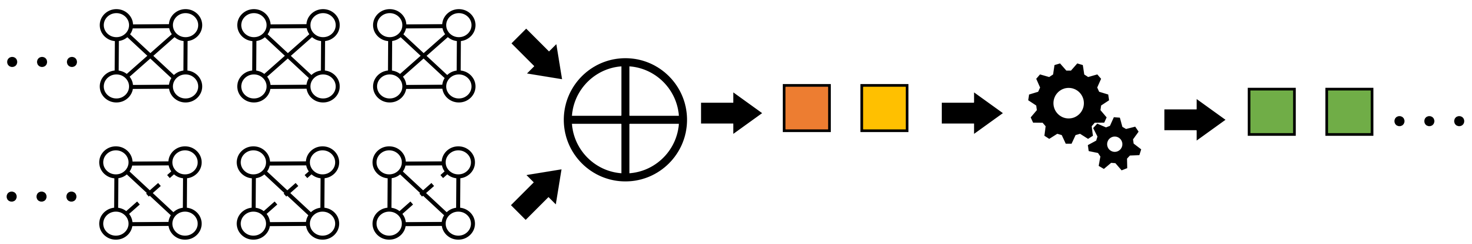


Result Transformation

- On-the-Fly
  - Match alternative patterns
  - Permute each match according to ϕ before applying UDF

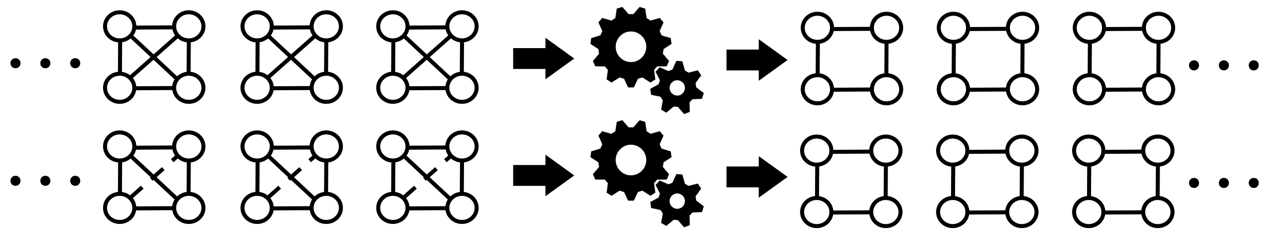


- Post-Matching
  - Match alternative patterns and apply UDF normally
  - Permute aggregation keys according to ϕ

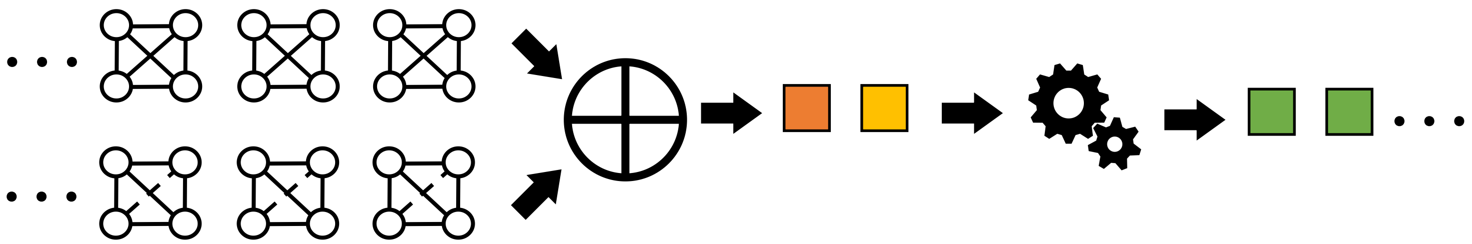


Result Transformation

- On-the-Fly
  - Match alternative patterns
  - Permute each match according to ϕ before applying UDF

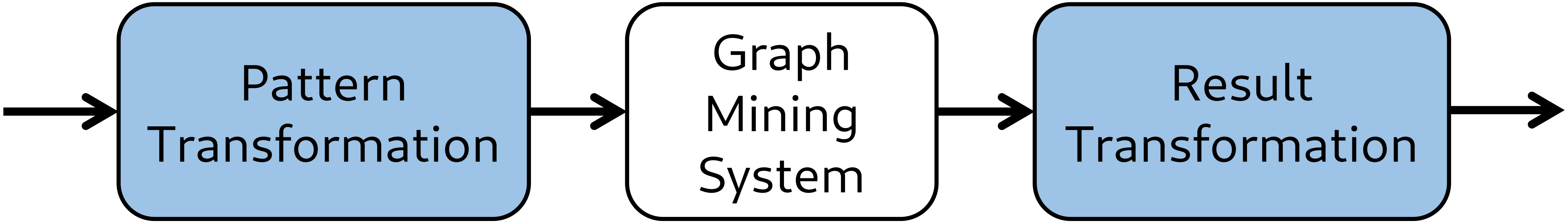


- Post-Matching
  - Match alternative patterns and apply UDF normally
  - Permute aggregation keys according to ϕ
  - Accumulate results



Subgraph Morphing

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



Evaluation

- 5 datasets

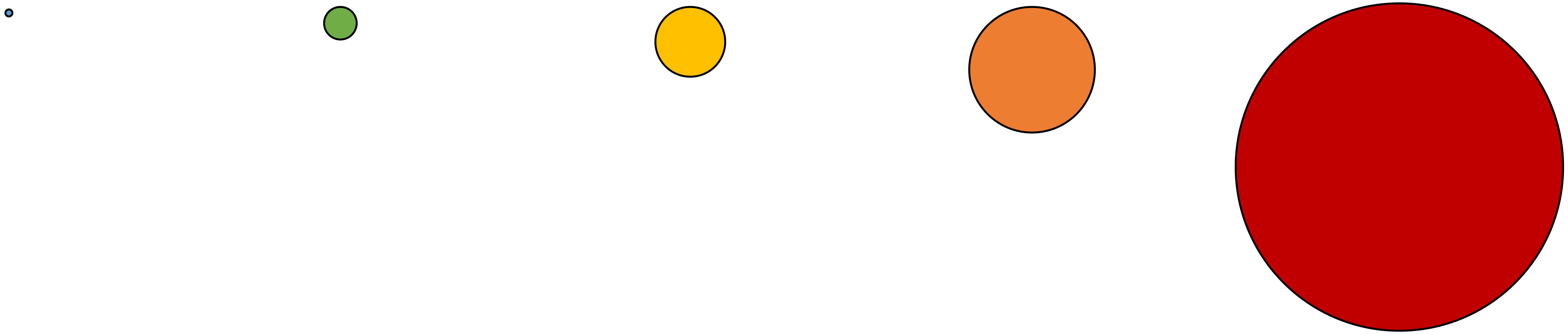
MiCo
|E| = 1M

MAG
|E| = 5.4M

Products
|E| = 61M

Orkut
|E| = 117M

Friendster
|E| = 1.8B



Evaluation

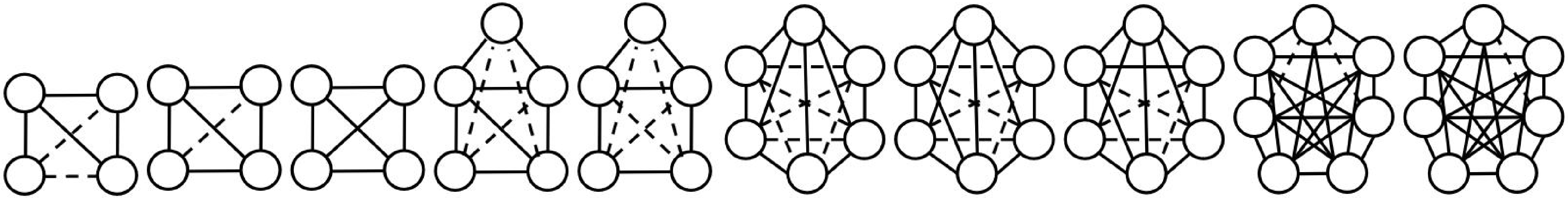
- 5 datasets
- 4 systems

Peregrine	[EuroSys ‘20]
AutoZero	[SOSP ‘19/OSR ‘21]
GraphPi	[SC ’20]
BiGJoin	[VLDB ‘18]

Evaluation

- 5 datasets
- 4 systems
- 21 pattern sets

- Different combinations of patterns
- Different variants of patterns
- {3,4}-FSM
- {3,4,5}-MC



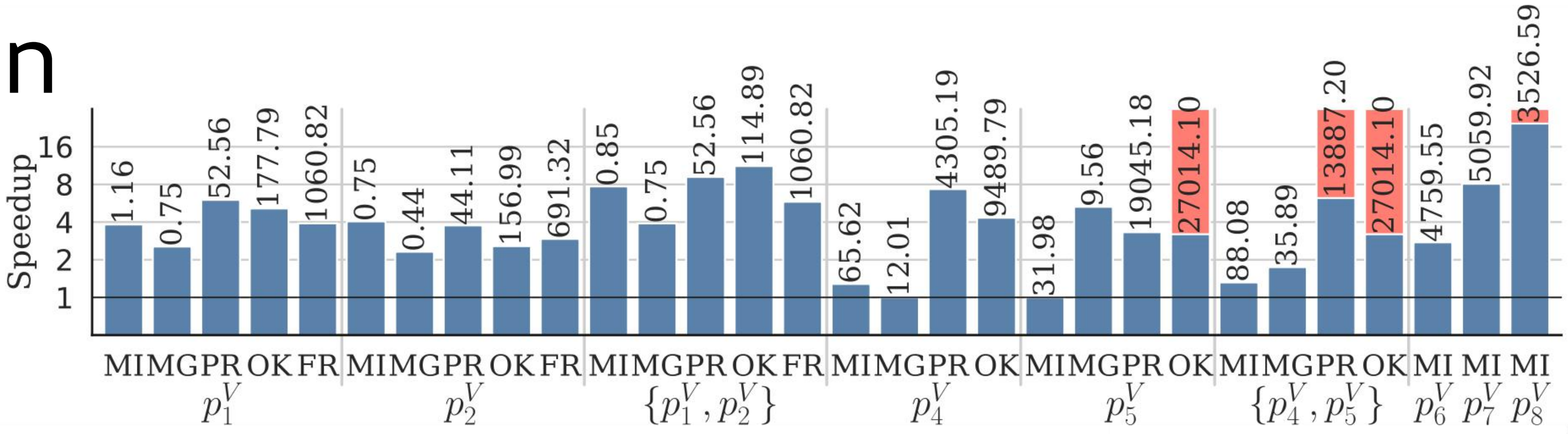
p_1^V p_2^V p_3 p_4^V p_5^V p_6^V p_7^V p_8^V p_9^V p_{10}^V

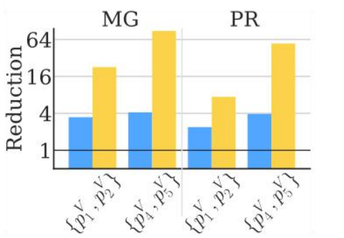
Evaluation

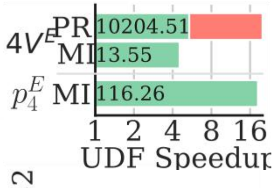
- 5 datasets
- 4 systems
- 21 pattern sets
- Execution time
- Bottleneck analysis

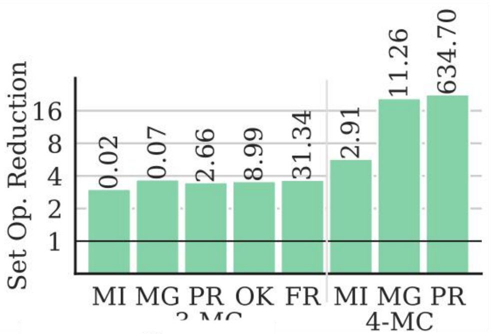
Evaluation

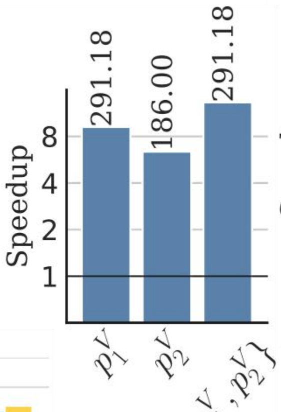
- 5 datasets
- 4 systems
- 21 pattern sets
- Execution time
- Bottleneck analysis

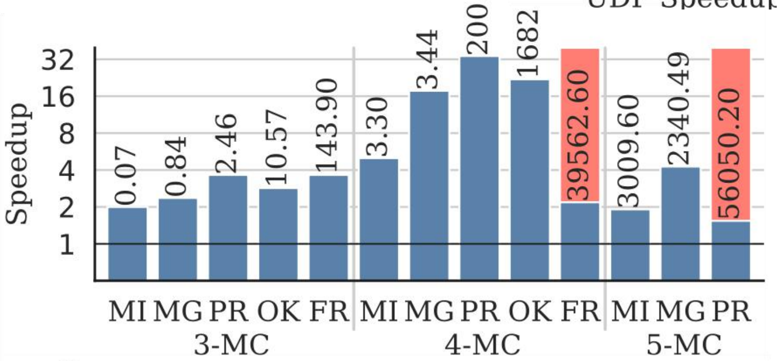


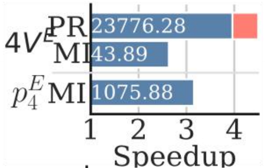


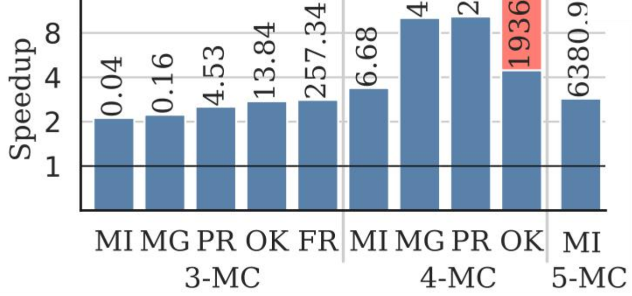


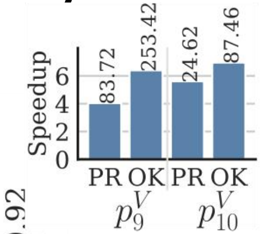


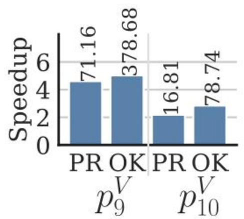


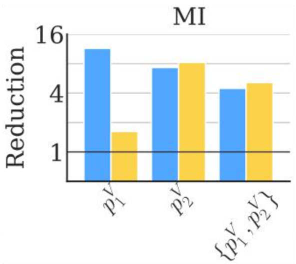


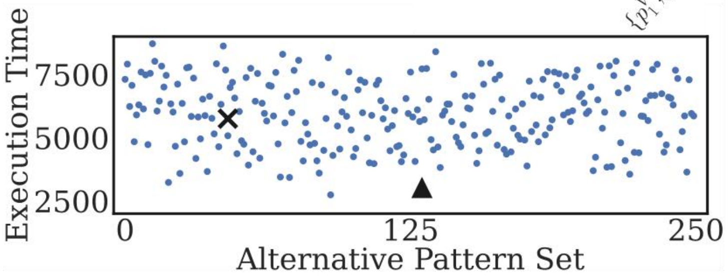


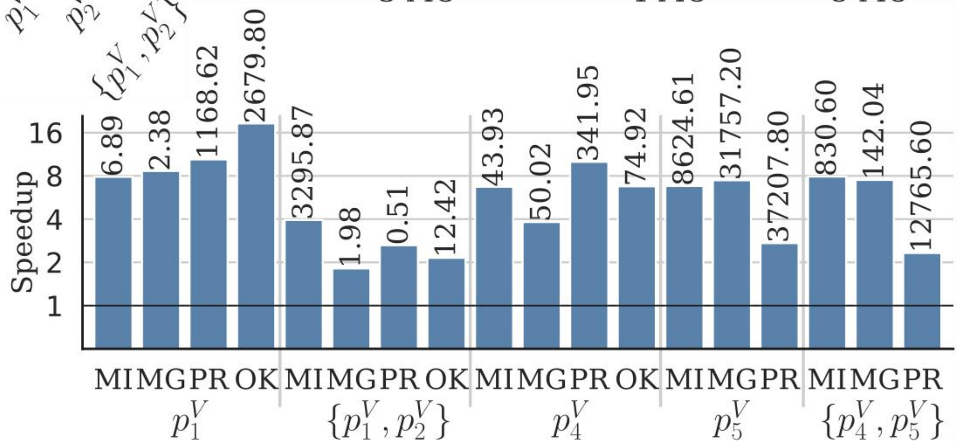








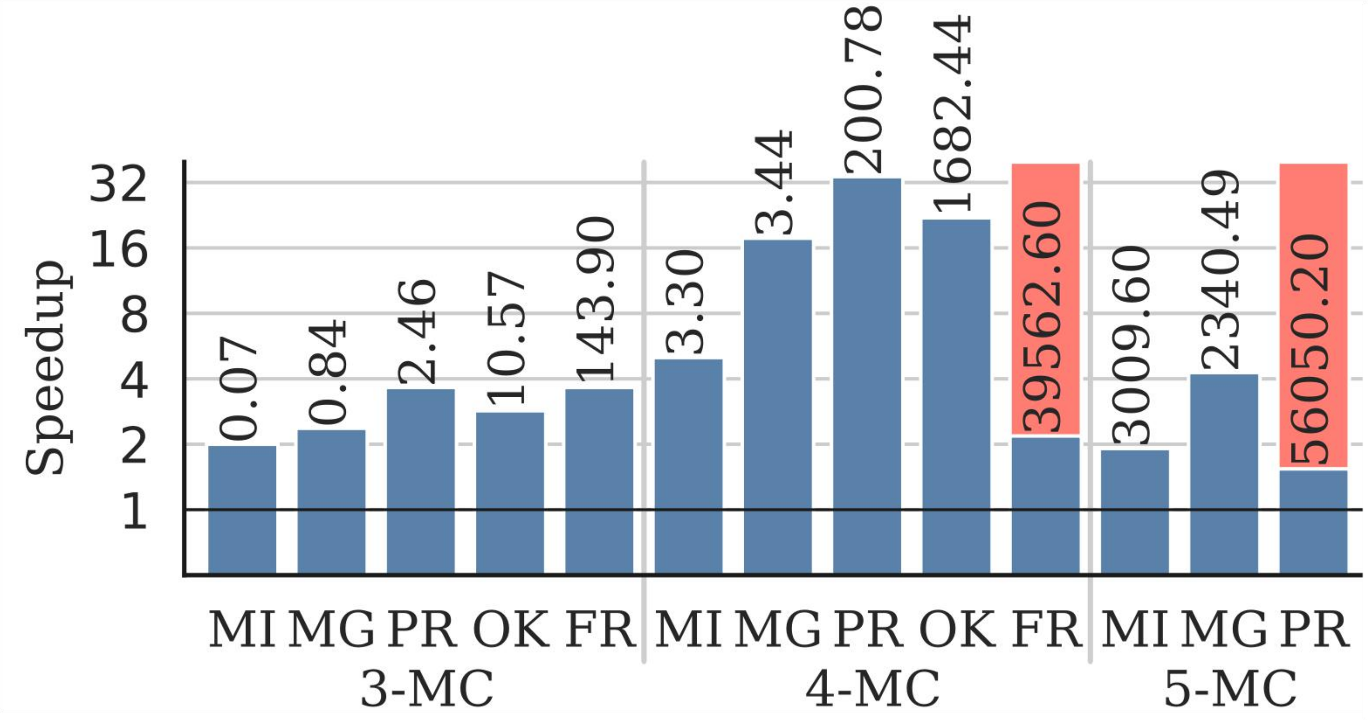




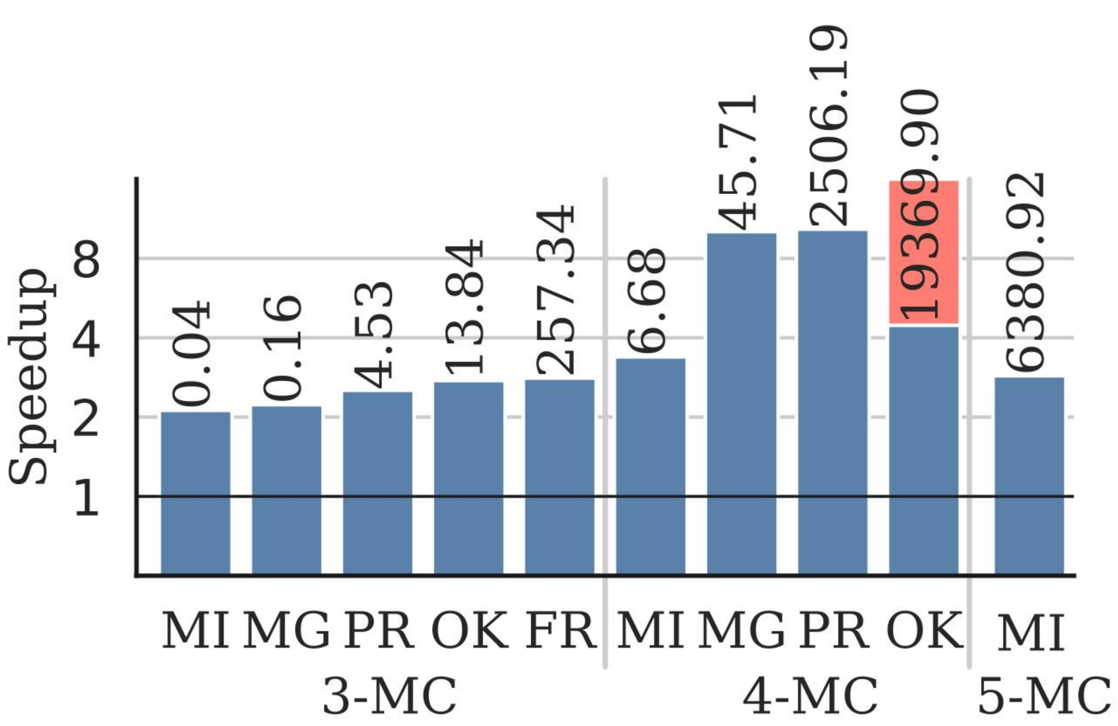
Evaluation: Highlights

- Best case → motif-counting 34x

Peregrine

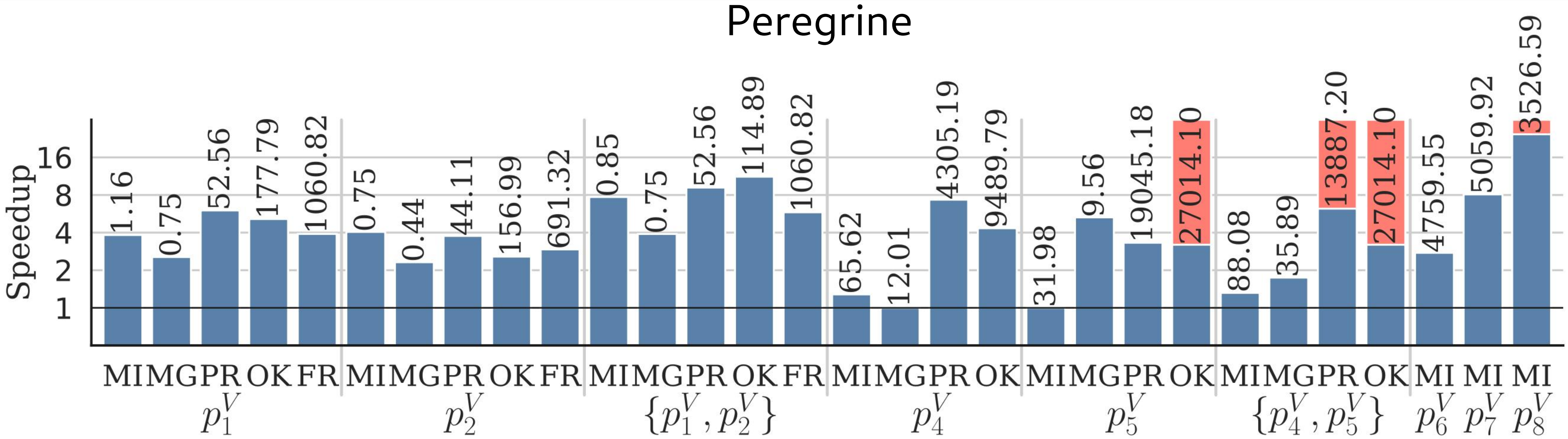


AutoZero



Evaluation: Highlights

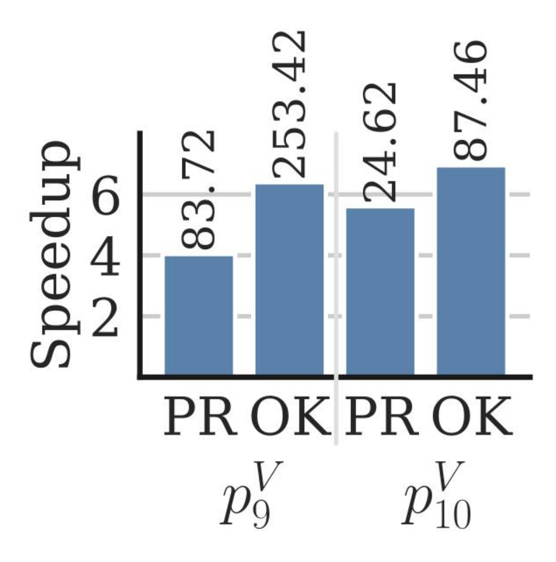
- Best case → motif-counting **34x** ⬆
- Worst case → single patterns **24x** ⬆



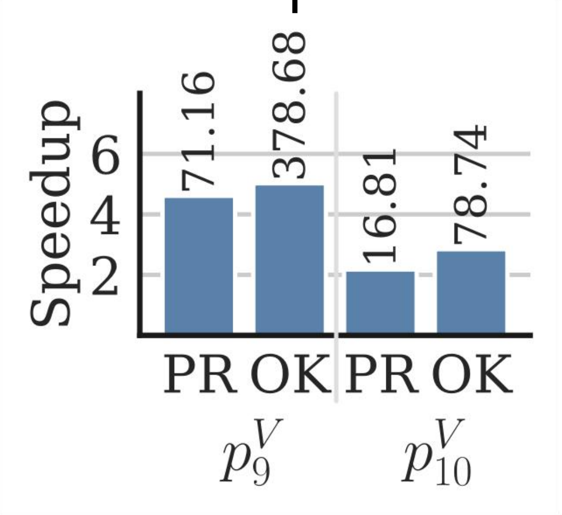
Evaluation: Highlights

- Best case → motif-counting **34x**
- Worst case → single patterns **24x**
- Large pattern enumeration **7x**

Peregrine

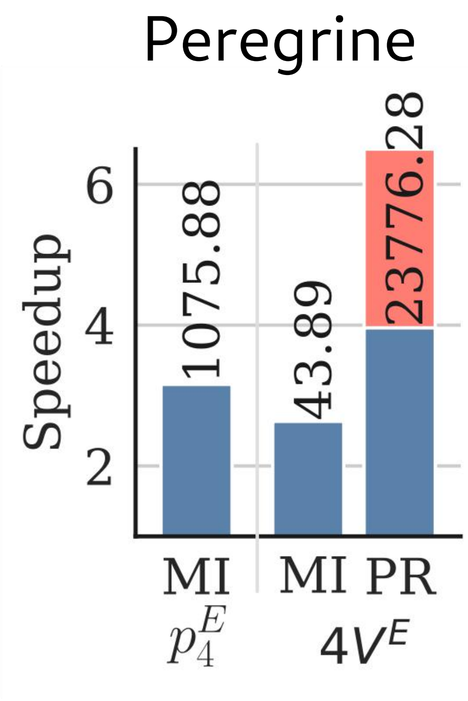


GraphPi



Evaluation: Highlights

- Best case → motif-counting **34x**
- Worst case → single patterns **24x**
- Large pattern enumeration **7x**
- On-the-fly conversion **4x**



Evaluation: Highlights

- Best case → motif-counting 34x ⬆
- Worst case → single patterns 24x ⬆
- Large pattern enumeration 7x ⬆
- On-the-fly conversion 4x ⬆

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

Thank You!

Natural Sciences and Engineering Research Council of Canada

SFU SIMON FRASER UNIVERSITY

PEREGRINE graphbolt lumos