CHALLENGING MEMORY AND TIME COMPLEXITY OF SUBGRAPH ISOMORPHISM PROBLEM WITH VF3
Challenging memory and time complexity of subgraph isomorphism problem with VF3
GRAPH MATCHING ALGORITHMS

- Exact Graph Matching
  - Structure Preserving Mapping
    - Adjacency-preserving
    - Non-adjacency preserving
  - NP-Complete Problem

Challenging memory and time complexity of subgraph isomorphism problem with VF3
SIMILARITY BY SUBGRAPH ISOMORPHISM

Challenging memory and time complexity of subgraph isomorphism problem with VF3
WHAT IS SUBGRAPH ISOMORPHISM?

- Given a graph:
  - Is it inside another graph?
  - How many times?
  - Where?

- Common applications:
  - Pattern search or Graph querying
  - Graph learning algorithms
  - Graph clustering

Challenging memory and time complexity of subgraph isomorphism problem with VF3
WHERE DID WE BEGIN FROM? VF2!


- VF2 is currently used in:
  - Boost C++ library
  - Networkx python library
  - MIT Courses

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: IN BRIEF

Inherited

- **State Space Representation**
  - Each state is partial solution
  - A goal is consistent complete solution

- **Depth-First** search with backtracking
  - State space as a tree by a total order relationship

- **Feasibility rules** to explore the space
  - Consistent states only
  - 2-levels Look-ahead

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: IN BRIEF

Inherited

▸ State Space Representation
  ▸ Each state is partial solution
  ▸ A goal is consistent complete solution

▸ Depth-First search with backtracking
  ▸ State space as a tree by a total order relationship

▸ Feasibility rules to explore the space
  ▸ Consistent states only
  ▸ 2-levels Look-ahead

New

▸ Pattern Pre-processing
  ▸ The first graph is explored before starting.

▸ Straightened look-ahead
  ▸ Node classification
    ▸ Structural and Semantic features

▸ New state transition function
  ▸ Significantly reduced the set where the next pairs are searched.
VF3: IN BRIEF

Inherited

- State Space Representation
  - Each state is partial solution
  - A goal is consistent complete solution

- Depth-First search with backtracking
  - State space as a tree by a total order relationship

- Feasibility rules to explore the space
  - Consistent states only
  - 2-levels Look-ahead

New

- Pattern Pre-processing
  - The first graph is explored before starting.

- Straightened look-ahead
  - Node classification
    - Structural and Semantic features

- New state transition function
  - Significantly reduced the set where the next pairs are searched.

Unchanged Complexity

- Linear in space
- Quadratic in time (avg)

But faster!
(From 10 to 1000 times)

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (WARMUP)

1. Classify the nodes
   ▶ Structural features
   ▶ Semantic features

2. Order the nodes
   ▶ Most constrained first
   ▶ Most rare first
   ▶ Generate a node sequence

3. Pre-process the pattern
   ▶ Prepare the structures
   ▶ Generate a coverage tree

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (EXPLORATION)

4. Exploring the state space

- Start from an empty solution
- Generate new states
  - Iteratively add a new matching couple by following the exploration sequence
- Check for the consistence
- Move through consistent states

(mapping)

\[ M(S_0) = \emptyset \]

\[ N = \{3, 1, 5, 2, 4\} \]

\[ G_1(S_0) \quad G_2(S_0) \]

Candidates of \( G_2 \)

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (EXPLORATION)

4. Exploring the state space

- Start from an empty solution
- Generate new states
  - Iteratively add a new matching couple by following the exploration sequence
- Check for the consistence
- Move through consistent state

State
- $S_0$
- $S_1$

Mapping
- $M(S_0) = \emptyset$
- $M(S_1) = \{(3, 3)\}$

$N = \{3, 1, 5, 2, 4\}$

Candidates of $G_2$

$G_1(S_1)$

$G_2(S_1)$

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (EXPLORATION)

4. Exploring the state space

- Start from an empty solution
- Generate new states
  - Iteratively add a new matching couple by following the exploration sequence
- Check for the consistence
- Move through consistent state

\[ \text{State} \quad S_0 \\
\text{Mapping} \\
M(S_0) = \emptyset \\
M(S_1) = \{(3, 3)\} \]

\[ \text{Candidates of } G_2 \]

\[ \text{N} = \{3, 1, 5, 2, 4\} \]

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (EXPLORATION)

4. Exploring the state space

- Start from an empty solution
- Generate new states
  - Iteratively add a new matching couple by following the exploration sequence
- Check for the consistence
- Move through consistent state

\[ N = \{3, 1, 5, 2, 4\} \]

\[ \begin{align*}
  G_1(S_0) &= \text{Candidates of } G_2 \\
  G_1(S_1) &= \{3, 3\} \\
  G_1(S_2) &= \{3, 3, (1, 6)\}
\end{align*} \]
VF3: HOW DOES IT WORK? (EXPLORATION)

4. Exploring the state space
   ▶ Start from an empty solution
   ▶ Generate new states
     ▶ Iteratively add a new matching couple by following the exploration sequence
   ▶ Check for the consistence
   ▶ Move through consistent state

State
S₀
S₁
S₂
S₃

Mapping
M(S₀) = Ø
M(S₁) = {(3, 3)}
M(S₂) = {(3, 3), (1, 6)}
M(S₃) = {(3, 3), (1, 6), (5, 1)}

N = {3, 1, 5, 2, 4}

Candidates of G₂

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (EXPLORATION)

4. Exploring the state space
   ▶ Start from an empty solution
   ▶ Generate new states
     ▶ Iteratively add a new matching couple by following the exploration sequence
     ▶ Check for the consistence
   ▶ Move through consistent state

\[
N = \{3, 1, 5, 2, 4\}
\]

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (EXPLORATION)

4. Exploring the state space
   ▶ Start from an empty solution
   ▶ Generate new states
     ▶ Iteratively add a new matching couple by following the exploration sequence
   ▶ Check for the consistence
   ▶ Move through consistent state

\[
\begin{align*}
N & = \{3, 1, 5, 2, 4\} \\
G_1(S_5) & = \begin{array}{c}
\begin{array}{c}
2 \\
B \\
3 \\
C \\
A \\
1 \\
D \\
5 \\
\end{array}
\end{array} \\
G_2(S_5) & = \begin{array}{c}
\begin{array}{c}
3 \\
C \\
4 \\
B \\
A \\
\end{array}
\end{array}
\end{align*}
\]

State | Mapping
---|---
\(S_0\) | \(M(S_0) = \emptyset\)
\(S_1\) | \(M(S_1) = \{(3, 3)\}\)
\(S_2\) | \(M(S_2) = \{(3, 3), (1, 6)\}\)
\(S_3\) | \(M(S_3) = \{(3, 3), (1, 6), (5, 1)\}\)
\(S_4\) | \(M(S_4) = \{(3, 3), (1, 6), (5, 1), (2, 2)\}\)
\(S_5\) | \(M(S_5) = \{(3, 3), (1, 6), (5, 1), (2, 4)\}\)

Candidates of \(G_2\)

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (EXPLORATION)

4. Exploring the state space

- Start from an empty solution
- Generate new states
  - Iteratively add a new matching couple by following the exploration sequence
- Check for the consistence
- Move through consistent state

\[ \text{State} \quad \text{Mapping} \]
\[ S_0 = \emptyset \quad M(S_0) = \emptyset \]
\[ S_1 \quad M(S_1) = \{(3, 3)\} \]
\[ S_2 \quad M(S_2) = \{(3, 3), (1, 6)\} \]
\[ S_3 \quad M(S_3) = \{(3, 3), (1, 6), (5, 1)\} \]
\[ S_4 \quad M(S_4) = \{(3, 3), (1, 6), (5, 1), (2, 2)\} \]
\[ S_5 \quad M(S_5) = \{(3, 3), (1, 6), (5, 1), (2, 4)\} \]
\[ S_6 \quad M(S_6) = \{(3, 3), (1, 6), (5, 1), (2, 4), (4, 2)\} \]

\[ N = \{3, 1, 5, 2, 4\} \]

Challenging memory and time complexity of subgraph isomorphism problem with VF3
How do we check for consistency?

Feasibility Rules
Challenging memory and time complexity of subgraph isomorphism problem with VF3

VF3: HOW DOES IT WORK? (FEASIBILITY)

How do we check for consistency?

Feasibility Rules

Core Rule

\{{{3,3},(1,6)}\}
VF3: HOW DOES IT WORK? (FEASIBILITY)

How do we check for consistency?

Feasibility Rules

Core Rule

Core Sets

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (FEASIBILITY)

How do we check for consistency?

Feasibility Rules

Core Rule

Semantic Inconsistency

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (FEASIBILITY)

How do we check for consistency?

Feasibility Rules

Core Rule

Semantic Inconsistence

Structural Inconsistence

Challenging memory and time complexity of subgraph isomorphism problem with VF3
How do we check for consistency?

Feasibility Rules

Look-ahead Rules

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (FEASIBILITY)

How do we check for consistency?

Feasibility Rules

Look-ahead Rules

Terminal Sets

Challenging memory and time complexity of subgraph isomorphism problem with VF3
VF3: HOW DOES IT WORK? (FEASIBILITY)

How do we check for consistency?

Feasibility Rules

Look-ahead Rules

Terminal Sets

Challenging memory and time complexity of subgraph isomorphism problem with VF3
Challenging memory and time complexity of subgraph isomorphism problem with VF3