Graph Similarity and Maximum Common Subtree Problem

Jeong Young Yun
Han Jin Il
Park Sung Soo
Contents

- Literature Review
- Problem Description (Maximum Common Subtree Problem)
- Degree Sequence Equivalence
- Algorithm
- Computational Result
- Conclusion
- References
Contents

➤ Literature Review
  ◆ Graph Similarity
  ◆ Isomorphism
  ◆ Maximum Common Subgraph

➤ Problem Description (Maximum Common Subtree Problem)

➤ Degree Sequence Equivalence

➤ Algorithm

➤ Computational Result

➤ Conclusion

➤ References
Graph Similarity

- Various applications: chemistry, biology, WWW, computer vision and pattern recognition
- Consider the graphs consist of a set of nodes and a set of edges
- Various kinds of notions
- Many kinds of variant problems
- Hard to handle in general

Figure 1: Protein networks of Bacteria and Yeast

Helicobacter pylori  Saccharomyces cerevisiae

Figure 1: Protein networks of Bacteria and Yeast
Literature Review

Isomorphism

Are these two graphs identical?

Graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there are bijections $\theta : V_1 \rightarrow V_2$ such that $uv \in E_1$ if and only if $\theta(u)\theta(v) \in E_2$ for all $u, v \in V_1$.

![Graph isomorphism example](image)

Figure 2: Graph isomorphism example

$\theta(a) = 1, \theta(b) = 4, \theta(c) = 2, \theta(d) = 5, \theta(e) = 3$
Literature Review

Maximum Common Subgraph

Are these graphs containing an identical subgraph?
How large is the size of this subgraph?

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, find subgraphs of $G_1$ and $G_2$ which are isomorphic each other with the maximum number of nodes or edges.

Figure 3: MCES and an MCIS example
Literature Review

Maximum Common Subgraph

Manić et al. [16]’s 0-1 integer programming formulation for the maximum common subgraph problem

maximize \[ \sum_{ij \in E_1} \sum_{kl \in E_2} z_{ijkl} \] subject to

\[ \sum_{k \in V_2} x_{ik} \leq 1, \quad \forall i \in V_1, \] (2)

\[ \sum_{i \in V_1} x_{ik} \leq 1, \quad \forall k \in V_2, \] (3)

\[ \sum_{kl \in E_2} z_{ijkl} \leq \sum_{k \in V_2} x_{ik}, \quad \forall ij \in E_1, \] (4)

\[ \sum_{ij \in E_1} z_{ijkl} \leq \sum_{i \in V_1} x_{ik}, \quad \forall kl \in E_2, \] (5)

\[ \sum_{j \in N(i)} z_{ijkl} \leq x_{ik} + x_{il}, \quad \forall i \in V_1, \forall kl \in E_2, \] (6)

\[ \sum_{l \in N(k)} z_{ijkl} \leq x_{ik} + x_{jk}, \quad \forall k \in V_2, \forall ij \in E_1, \] (7)

Decision Variables:

\[ z_{ijkl} = \begin{cases} 1, & \text{if } ij \in E_1 \text{ is matched to } kl \in E_2, \\ 0, & \text{otherwise.} \end{cases} \]

\[ x_{ik} = \begin{cases} 1, & \text{if } i \in V_1 \text{ is matched to } k \in V_2, \\ 0, & \text{otherwise.} \end{cases} \]

Restrictions:

Node assignment requirements: (2) and (3)

Edge assignment requirements: (4) and (5)

Isomorphism requirements: (6) and (7)

Binary variables: (8)
Contents

➢ Literature Review

➢ Problem Description (Maximum Common Subtree Problem)
   - Definition of the Maximum Common Subtree Problem
   - Basic Ideas

➢ Degree Sequence Equivalence

➢ Algorithm

➢ Computational Result

➢ Conclusion

➢ References
Problem Description

Definition of the Maximum Common Subtree problem

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, find subgraphs of $G_1$ and $G_2$ having tree structure which are isomorphic each other with the maximum number of nodes.

Graphs are assumed to be simple, connected and non-trivial.

Figure 4 : MCST example
Problem Description

Figure 5: MCST example
Problem Description

Basic Ideas

Two graphs are isomorphic $\rightarrow$ Degree Sequences are same

Directly Finding Maximum Common Subgraph (MCS) is not easy

We do Not Use!

Finding Degree Sequence Equivalent Subgraph (DSES) is relatively easier than MCS &
Checking the Tree Isomorphism is easy

We Use!

Strategy

Check the Tree Isomorphism whenever the algorithm detects the DSES having tree structure!
Contents

➤ Literature Review

➤ Problem Description (Maximum Common Subtree Problem)

➤ Degree Sequence Equivalence
  ◯ Terminologies and Definition
  ◯ Formulation for the maximum degree sequence equivalent subgraph problem (MDSES)

➤ Algorithm

➤ Computational Result

➤ Conclusion

➤ References
Degree Sequence Equivalence

Terminologies and Definition

- (Node) Degree
  - The number of edges that are incident with a particular node

- Degree Sequence
  - The set of node degree which is a sorted sequence in non-increasing order

- Maximum Degree Sequence Equivalent Subgraph problem
  - Given two graphs \( G_1 = (V_1, E_1) \) and \( G_2 = (V_2, E_2) \), find subgraphs of \( G_1 \) and \( G_2 \) (not necessary isomorphic) having the same degree sequences with the maximum total degree

Figure 6: The Königsberg graph having a degree sequence (5, 3, 3, 3)
Degree Sequence Equivalence

The maximum degree sequence equivalent subgraph problem (MDSES)

Decision Variables:

\[ x_{ik} = \begin{cases} 1, & \text{if } i \in V_1 \text{ is matched to } k \in V_2, \\ 0, & \text{otherwise.} \end{cases} \]  

\[ y_{ij} = \begin{cases} 1, & \text{if } ij \in E_1 \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases} \]  

\[ z_{kl} = \begin{cases} 1, & \text{if } kl \in E_2 \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases} \]

Restrictions:

Node assignment requirements: (2) and (3)

Degree sequence equivalence requirements: (4)

\[ x_{ik} = 1 \Rightarrow \sum_{j \in N(i)} y_{ij} = \sum_{l \in N(k)} z_{kl}, \]

\[ x_{ik} = 0 \Rightarrow \text{Unconstrained.} \]

Binary variables: (5)

Where \( d_{\text{max}} \) is the maximum node degree of given graphs.
Degree Sequence Equivalence

» MDSES example

Degree Sequence : \( (4, 4, 2, 2, 2, 2, 1, 1, 1, 1) \)
Matched node pairs : \( \{ (1, 1), (4, 4), (0, 8), (3, 5), (6, 3), (8, 2), (2, 9), (5, 6), (7, 0), (9, 7) \} \)

Figure 7: MDSES example
Contents

➢ Literature Review
➢ Problem Description (Maximum Common Subtree Problem)
➢ Degree Sequence Equivalence
➢ Algorithm
  ➢ Flowchart (in brief)
  ➢ Formulation Decomposition Technique
  ➢ Checking the Tree Isomorphism
  ➢ Flowchart (in detail)
➢ Computational Result
➢ Conclusion
➢ References
Algorithm

Flowchart (in brief)

Figure 8 : Flowchart of our algorithm
Algorithm

Mathematical Formulation for finding the DSES having tree structures

\[
\text{maximize} \quad \sum_{ij \in E_1} y_{ij} \\
\text{subject to} \\
\sum_{k \in V_2} x_{ik} = 1, \quad \forall i \in V_1, \\
\sum_{i \in V_1} x_{ik} = 1, \quad \forall k \in V_2, \\
\left| \sum_{j \in N(i)} y_{ij} - \sum_{l \in N(k)} z_{kl} \right| \leq d_{\max}(1 - x_{ik}), \quad \forall i \in V_1, \forall k \in V_2, \\
\sum_{ij \in E_1} y_{ij} = T_s - 1, \\
\sum_{kl \in E_2} z_{kl} = T_s - 1, \\
\sum_{ij \in E(S_1)} y_{ij} \leq |S_1| - 1, \quad S_1 \subset V_1, S_1 \neq \emptyset, V_1, \\
\sum_{ij \in E(S_2)} z_{kl} \leq |S_2| - 1, \quad S_2 \subset V_2, S_2 \neq \emptyset, V_2, \\
x, y, z \text{ binary.}
\]

Decision Variables:

\[
x_{ik} = \begin{cases} 
1, & \text{if } i \in V_1 \text{ is matched to } k \in V_2, \\
0, & \text{otherwise.}
\end{cases}
\]

\[
y_{ij} = \begin{cases} 
1, & \text{if } ij \in E_1 \text{ is selected,} \\
0, & \text{otherwise.}
\end{cases}
\]

\[
z_{kl} = \begin{cases} 
1, & \text{if } kl \in E_2 \text{ is selected,} \\
0, & \text{otherwise.}
\end{cases}
\]

Restrictions:

Node assignment requirements : (2) and (3)
Degree sequence equivalence requirements: (4)
Tree structure requirements : (5),(6),(7) and (8)
Binary variables: (9)
Algorithm

Formulation Decomposition Technique

- **Master**: 
  
  \[
  \begin{align*}
  \text{maximize} & \quad 0 \\
  \text{subject to} & \quad \sum_{k \in V_2} x_{ik} = 1, \quad \forall i \in V_1, \quad (1) \\
  & \quad \sum_{i \in V_1} x_{ik} = 1, \quad \forall k \in V_2, \quad (2) \\
  & \quad x \text{ binary.} \quad (3)
  \end{align*}
  \]
  
  Master solution: Node matching between two graphs
  
  Combinatorial Benders’ cuts [8] are recursively added

- **Slave** (\(\tilde{x}\)), a linear system parameterized by \(\tilde{x}\): 
  
  \[
  \begin{align*}
  \text{maximize} & \quad \sum_{ij \in E_1} y_{ij} \\
  \text{subject to} & \quad \sum_{j \in N(i)} y_{ij} - \sum_{l \in N(k)} z_{kl} = 0, \quad \forall i \in V_1, \forall k \in V_2, \quad (5) \\
  & \quad \sum_{ij \in E_1} y_{ij} = T_s - 1, \quad (6) \\
  & \quad \sum_{k \in E_2} z_{kl} = T_s - 1, \quad (7) \\
  & \quad \sum_{ij \in E(S_1)} y_{ij} \leq |S_1| - 1, \quad S_1 \subset V_1, S_1 \neq \emptyset, V_1, \quad (9) \\
  & \quad \sum_{k \in E(S_2)} z_{kl} \leq |S_2| - 1, \quad S_2 \subset V_2, S_2 \neq \emptyset, V_2, \quad (10) \\
  & \quad y, z \text{ binary.} \quad (11)
  \end{align*}
  \]
  
  Slave solution: Two subtrees from the given graphs
  
  (9) and (10) are handled by Branch-&-Cut approach
Algorithm

Checking the Tree Isomorphism

AHU-algorithm [1] ~ O(n)

Figure 9: Checking the tree isomorphism by using the AHU algorithm
Algorithm

Flowchart (in detail)

Figure 10: Flowchart of our algorithm
Contents

- Literature Review
- Problem Description (Maximum Common Subtree Problem)
- Degree Sequence Equivalence
- Algorithm
- Computational Result
- Conclusion
- References
Computational Result

- Experimental Setting
  - Programming Language: C++
  - Package: ILOG CPLEX 11.0 (Optimization), MATLAB 7.0 (Graph generating)
  - O/S: Microsoft Windows 7
  - H/W: PC AMD Athlon™ 64 X2 Dual Core Processor 5600+ 2.90GHz with 4GB RAM

- Input Data
  - Randomly generated graphs by Erdős and Rényi model ((n, p) model)
  - Randomly generated trees
  - Scale-free networks
  - Consider the graphs having 10, 20 and 30 nodes
## Computational Result

<table>
<thead>
<tr>
<th>G₁</th>
<th>G₂</th>
<th>Manić et al.’s algorithm</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>V₁</td>
<td>E₁</td>
<td>V₂</td>
<td>E₂</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>20</td>
<td>37</td>
<td>20</td>
<td>42</td>
</tr>
<tr>
<td>20</td>
<td>36</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>20</td>
<td>41</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>30</td>
<td>85</td>
</tr>
<tr>
<td>30</td>
<td>76</td>
<td>30</td>
<td>74</td>
</tr>
<tr>
<td>30</td>
<td>91</td>
<td>30</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 1: Comparison of the two algorithms for the random graphs when p=0.2

<table>
<thead>
<tr>
<th>G₁</th>
<th>G₂</th>
<th>Manić et al.’s algorithm</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>V₁</td>
<td>E₁</td>
<td>V₂</td>
<td>E₂</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>80</td>
<td>20</td>
<td>72</td>
</tr>
<tr>
<td>20</td>
<td>84</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td>20</td>
<td>82</td>
<td>20</td>
<td>80</td>
</tr>
<tr>
<td>30</td>
<td>177</td>
<td>30</td>
<td>173</td>
</tr>
<tr>
<td>30</td>
<td>173</td>
<td>30</td>
<td>171</td>
</tr>
<tr>
<td>30</td>
<td>179</td>
<td>30</td>
<td>177</td>
</tr>
</tbody>
</table>

Table 2: Comparison of the two algorithms for the random graphs when p=0.4
## Computational Result

<table>
<thead>
<tr>
<th>G₁</th>
<th>G₂</th>
<th>Manić et al.'s algorithm</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Run time(s)</td>
<td>Solution</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>0.1</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>0.1</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>0.1</td>
<td>9</td>
</tr>
<tr>
<td>20</td>
<td>80</td>
<td>42.7</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>72</td>
<td>85.2</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>84</td>
<td>113.8</td>
<td>19</td>
</tr>
<tr>
<td>30</td>
<td>177</td>
<td>600.0*</td>
<td>28*</td>
</tr>
<tr>
<td>30</td>
<td>173</td>
<td>543.2</td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td>173</td>
<td>600.0*</td>
<td>27*</td>
</tr>
</tbody>
</table>

Table 3: Comparison of the two algorithms for the random graph and the random tree with k=2

<table>
<thead>
<tr>
<th>G₁</th>
<th>G₂</th>
<th>Manić et al.'s algorithm</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Run time(s)</td>
<td>Solution</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>0.2</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>0.1</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>0.2</td>
<td>7</td>
</tr>
<tr>
<td>20</td>
<td>80</td>
<td>112.7</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>72</td>
<td>215.1</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>84</td>
<td>343.0</td>
<td>19</td>
</tr>
<tr>
<td>30</td>
<td>177</td>
<td>600.0*</td>
<td>23*</td>
</tr>
<tr>
<td>30</td>
<td>173</td>
<td>600.0*</td>
<td>24*</td>
</tr>
<tr>
<td>30</td>
<td>179</td>
<td>600.0*</td>
<td>26*</td>
</tr>
</tbody>
</table>

Table 4: Comparison of the two algorithms for the random graph and the random tree with k=unbounded
## Computational Result

<table>
<thead>
<tr>
<th></th>
<th>G₁</th>
<th>G₂</th>
<th>Manić et al.’s algorithm</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run time(s)</td>
<td>Solution</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>10</td>
<td>1.2</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>10</td>
<td>1.4</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>10</td>
<td>2.0</td>
<td>9</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>20</td>
<td>198.4</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>33</td>
<td>20</td>
<td>244.1</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>29</td>
<td>20</td>
<td>87.5</td>
<td>19</td>
</tr>
<tr>
<td>30</td>
<td>38</td>
<td>30</td>
<td>600.0*</td>
<td>27*</td>
</tr>
<tr>
<td>30</td>
<td>55</td>
<td>30</td>
<td>600.0*</td>
<td>27*</td>
</tr>
<tr>
<td>30</td>
<td>46</td>
<td>30</td>
<td>600.0*</td>
<td>20*</td>
</tr>
</tbody>
</table>

Table 5: Comparison of the two algorithms for the scale-free networks

- Our algorithm performs relatively well compared to the revised Manić et al.’s algorithm
Contents

- Literature Review
- Problem Description (Maximum Common Subtree Problem)
- Degree Sequence Equivalence
- Algorithm
- Computational Result
- Conclusion
  - Summary
  - Further Research
- References
Conclusion

Summary

- In order to find the maximum common subtree, our algorithm uses the integer programming model to find the degree sequence equivalent subgraph and AHU-algorithm \[1\] to check the tree isomorphism.
- To solve the integer programming model, we use decomposition technique \[8\] and apply Branch-&-Cut method.
- Our algorithm performs relatively well in solving of random instances of moderate size compared to the revised Manić’s approach \[16\].
Conclusion

Further research

- Further researches are needed for the case of another special graphs such as directed trees, forests and planar graphs.
- Complexity analysis is needed via further experiments.
- There is much room for the improvement of the formulation to find the degree sequence equivalent subgraph problem.
- Studies for the strengthening of the bound for the maximum common subgraph problem are needed.
Contents

➢ Literature Review
➢ Problem Description (Maximum Common Subtree Problem)
➢ Degree Sequence Equivalence
➢ Algorithm
➢ Computational Result
➢ Conclusion
➢ References
References


References


Q/A

Systems Optimization

Jeong Young Yun
Supplements

Revision of Manić et.al’s approach for MCST

- Add inequalities representing the tree structure requirements
  \[ \sum_{ij \in E_1} \sum_{kl \in E_2} z_{ijkl} = T_s - 1, \quad (1) \]
  \[ \sum_{ij \in E(S_1)} \sum_{kl \in E_2} z_{ijkl} \leq |S_1| - 1, \quad S_1 \subset V_1, S_1 \neq \emptyset, V_1, \quad (2) \]
  \[ \sum_{ij \in E_1} \sum_{kl \in E(S_2)} z_{ijkl} \leq |S_2| - 1, \quad S_2 \subset V_2, S_2 \neq \emptyset, V_2, \quad (3) \]

Where \( T_s \) is the number of nodes in tree that we want to find.

- Recursively solve the revised formulation until we find the maximum common subgraph having the tree structure

- Poor performance: barely solve instance having 25 nodes
Supplements

Adjusting the tree size

To find the common subtree with the “maximum number of nodes”

Recursively solve the formulation to detect the DSES with the fixed number of nodes

Adjusting rule

- At first, find the DSES having Ts nodes
- If we find the isomorphic tree, then stop. o/w Ts’ is adjusted as follows
  - Set Ts’=[Ts/2] at first, o/w Ts’= [Ts’/2]
  - If the formulation is feasible, then set Ts=Ts+Ts’
  - If the formulation is infeasible, then set Ts=Ts-Ts’
  - When Ts’=0, stop
Adjusting the tree size

Set $T_s = 5$

When $T_s = 5$, we can not find the common subtree

Set $T_s = T_s - T_s' = 5 - \lfloor 5/2 \rfloor = 3$

When $T_s = 3$, we can find the common subtree

Set $T_s = T_s + T_s' = 5 + \lfloor 2/2 \rfloor = 4$

When $T_s = 4$, we can find the common subtree

Since $T_s' = \lfloor 1/2 \rfloor = 0$, the algorithm terminates

Figure 11: Example of adjusting the tree size