대한산업공학회/한국경영과학회
2010년 춘계공동학술대회

Graph Similarity and Maximum Common Subtree Problem

Jeong Young Yun
Han Jin Il
Park Sung Soo

ISysE KAIST
Industrial & Systems Engineering

Contents

Contents

Literature Review

- 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

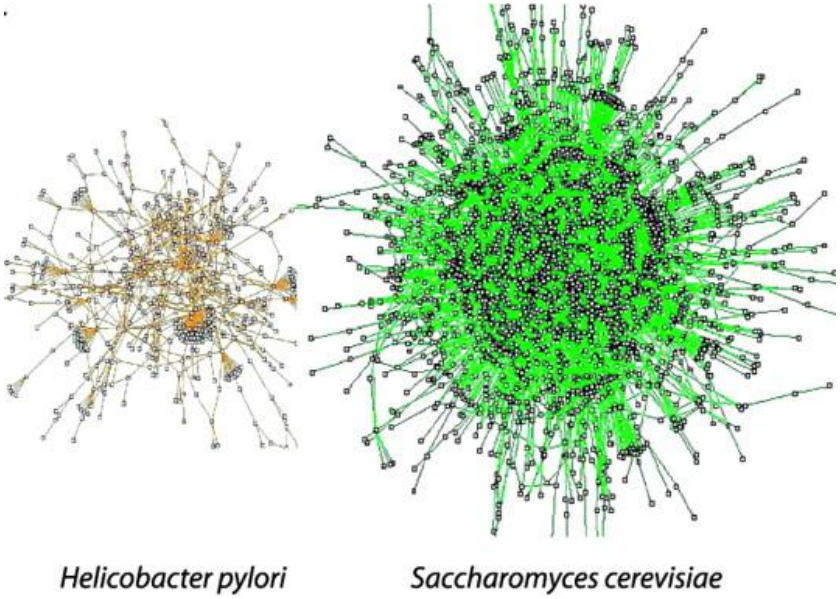


Figure 1: Protein networks of Bacteria and Yeast

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 $\theta : V_1 \to V_2$ such that $uv \in E_1$ if and only if $\theta(u)\theta(v) \in E_2$ for all $u, v \in V_1$

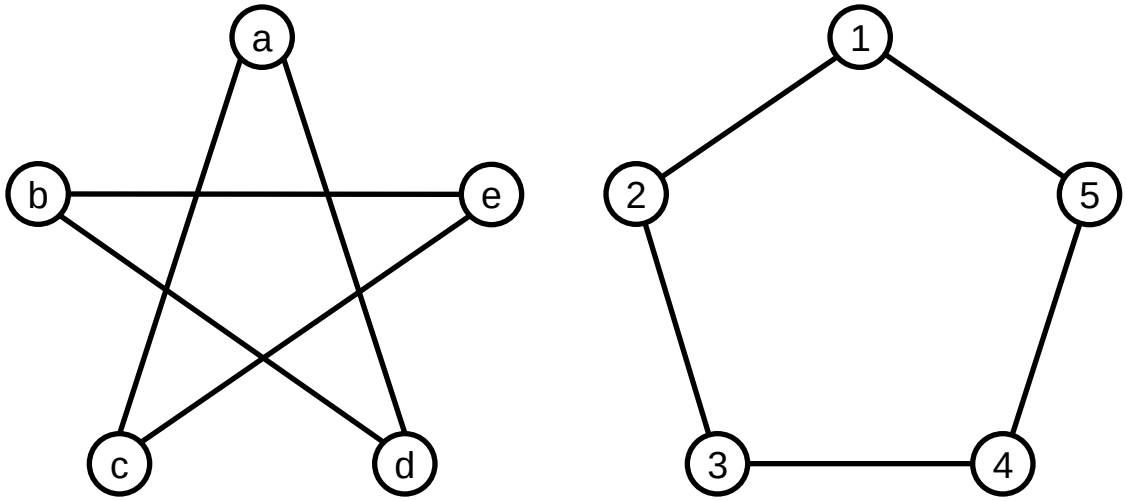


Figure 2: Graph isomorphism example

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

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

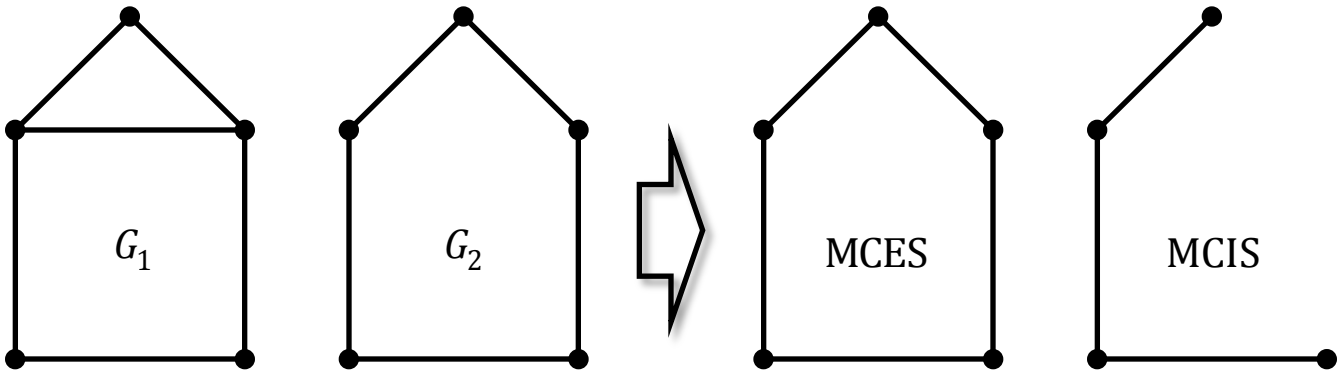


Figure 3 : MCES and an MCIS example

Literature Review

Maximum Common Subgraph

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

$$\text{maximize} \quad \sum_{ij \in E_1} \sum_{kl \in E_2} z_{ijkl} \qquad (1)$$

subject to

$$\sum_{k \in V_2} x_{ik} \le 1, \qquad \forall i \in V_1, \qquad (2)$$

$$\sum_{i \in V_1} x_{ik} \le 1, \qquad \forall k \in V_2, \qquad (3)$$

$$\sum_{kl \in E_2} z_{ijkl} \le \sum_{k \in V_2} x_{ik}, \qquad \forall ij \in E_1, \qquad (4)$$

$$\sum_{ij \in E_1} z_{ijkl} \le \sum_{i \in V_1} x_{ik}, \qquad \forall kl \in E_2, \qquad (5)$$

$$\sum_{j \in N(i)} z_{ijkl} \le x_{ik} + x_{il}, \qquad \forall i \in V_1, \forall kl \in E_2, \qquad (6)$$

$$\sum_{l \in N(k)} z_{ijkl} \le x_{ik} + x_{jk}, \qquad \forall k \in V_2, \forall ij \in E_1, \qquad (7)$$

$$x, z \quad \text{binary}. \qquad (8)$$

- 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

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

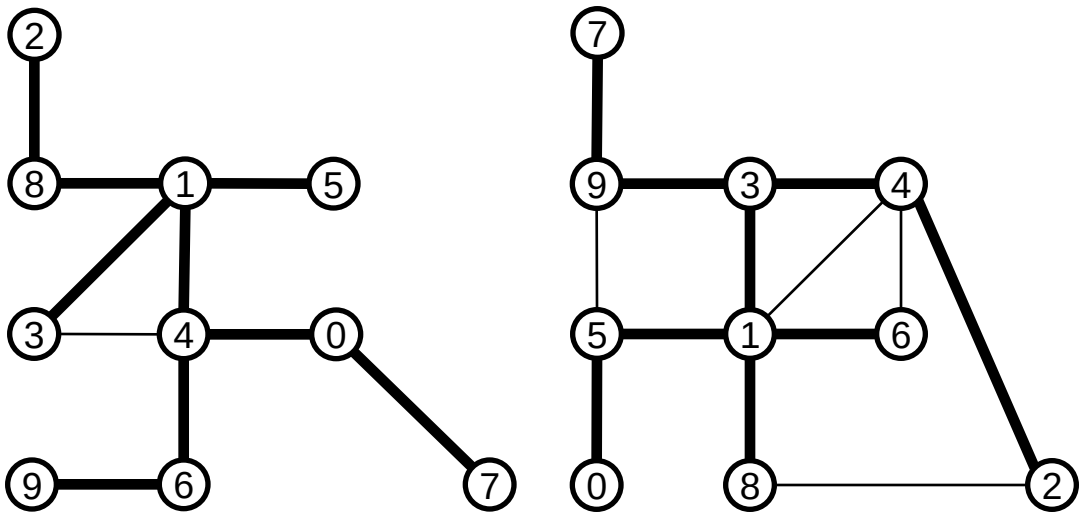


Figure 4 : MCST example

Problem Description

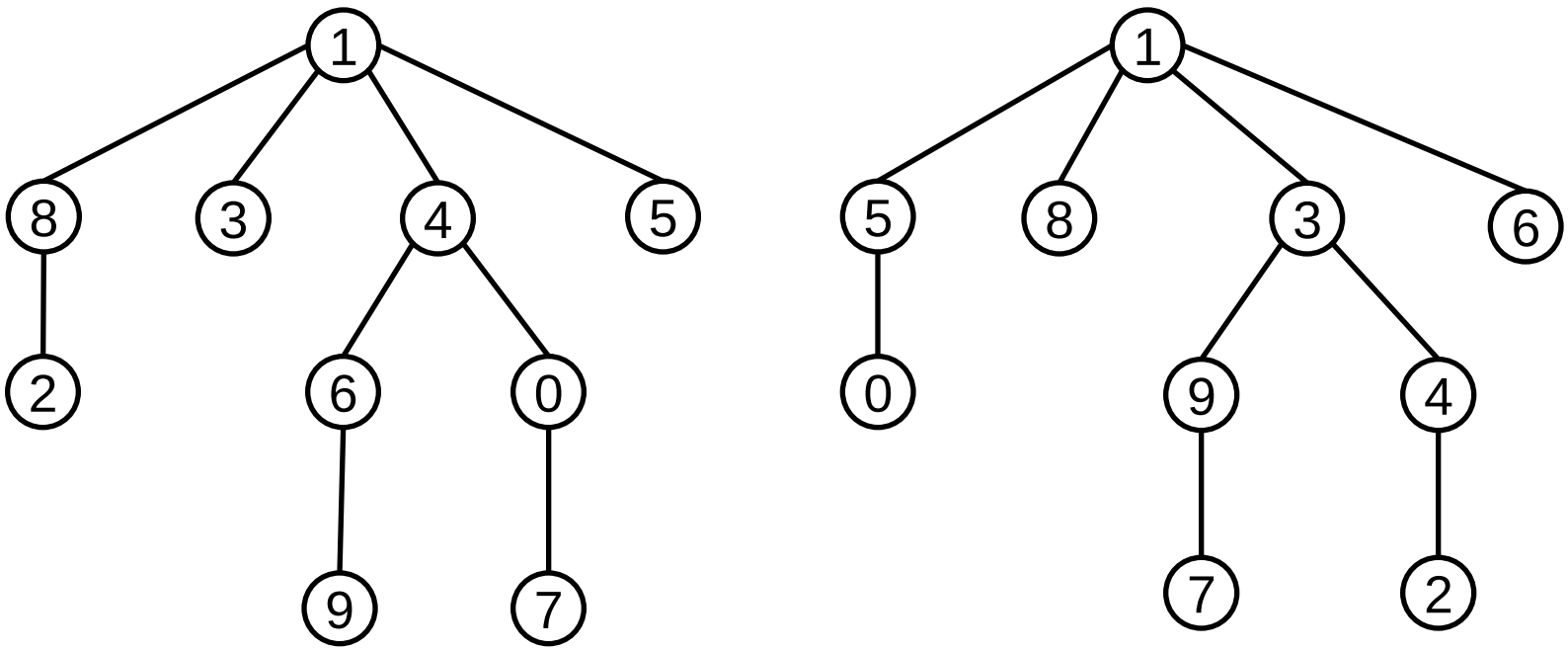


Figure 5 : MCST example

Problem Description

Basic Ideas

Two graphs are isomorphic ⟶ 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

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

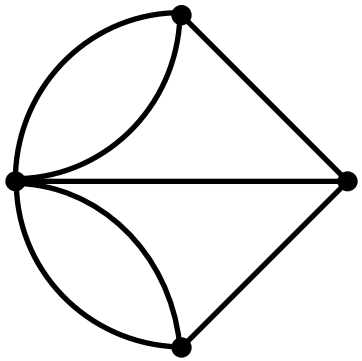


Figure 6 : The Königsberg graph having a degree sequence (5, 3, 3, 3)

Degree Sequence Equivalence

- The maximum degree sequence equivalent subgraph problem (MDSES)

$$\text{maximize} \quad \sum_{ij \in E_1} y_{ij} \qquad (1)$$

subject to

$$\sum_{k \in V_2} x_{ik} = 1, \qquad \forall i \in V_1, \qquad (2)$$

$$\sum_{i \in V_1} x_{ik} = 1, \qquad \forall k \in V_2, \qquad (3)$$

$$\left| \sum_{j \in N(i)} y_{ij} - \sum_{l \in N(k)} z_{kl} \right| \le d_{\max}(1 - x_{ik}), \quad \forall i \in V_1, \forall k \in V_2, \qquad (4)$$

$$x, y, z \ \text{ binary}, \qquad (5)$$

Where $d_{\max}$ is the maximum node degree of given graphs.

- 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)

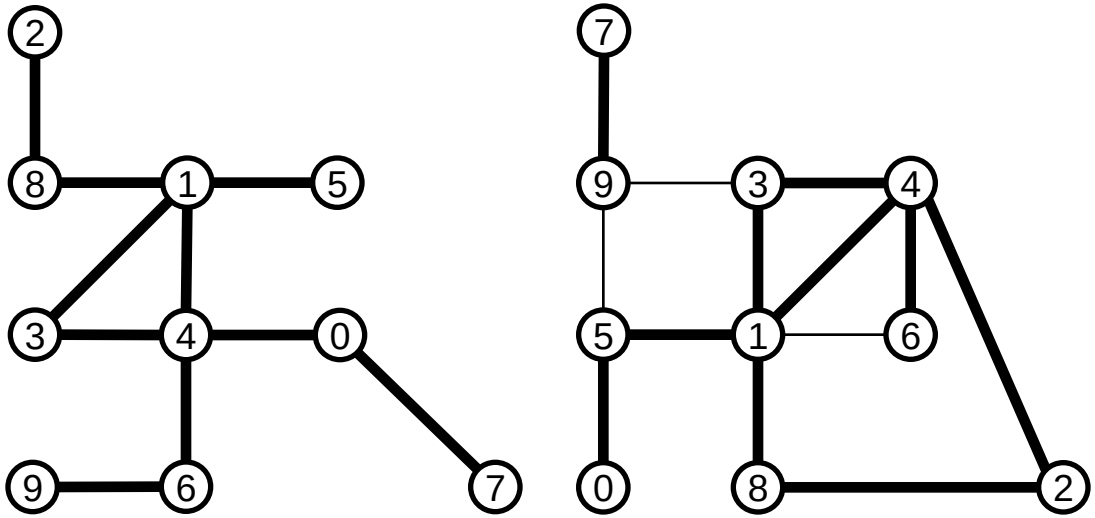
Degree sequence equivalence requirements: (4)

$$\begin{cases} x_{ik} = 1 \Rightarrow \sum_{j \in N(i)} y_{ij} = \sum_{l \in N(k)} z_{kl}, \\ x_{ik} = 0 \Rightarrow \text{Unconstrained.} \end{cases}$$

Binary variables: (5)

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

Algorithm

- Flowchart (in brief)

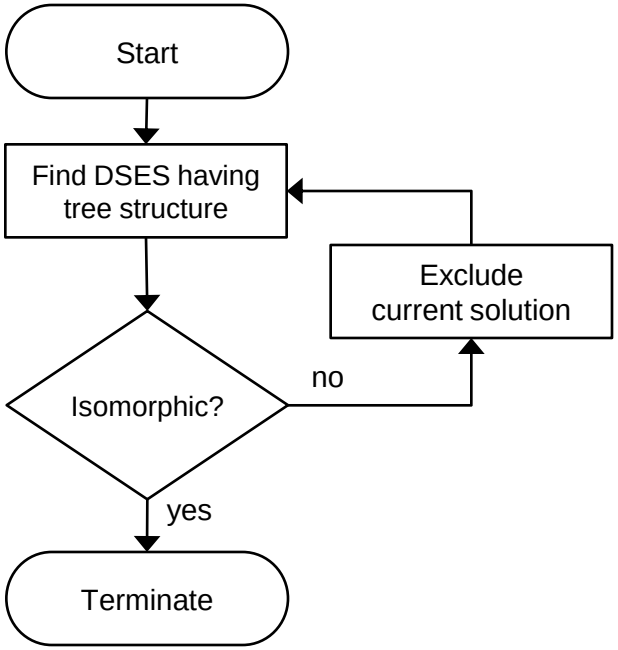


Figure 8 : Flowchart of our algorithm

Algorithm

- Mathematical Formulation for finding the DSES having tree structures

$$\text{maximize} \quad \sum_{ij \in E_1} y_{ij} \tag{1}$$

subject to

$$\sum_{k \in V_2} x_{ik} = 1, \qquad \forall i \in V_1, \tag{2}$$

$$\sum_{i \in V_1} x_{ik} = 1, \qquad \forall k \in V_2, \tag{3}$$

$$\left| \sum_{j \in N(i)} y_{ij} - \sum_{l \in N(k)} z_{kl} \right| \le d_{\max}(1 - x_{ik}), \quad \forall i \in V_1, \forall k \in V_2, \tag{4}$$

$$\sum_{ij \in E_1} y_{ij} = T_s - 1, \tag{5}$$

$$\sum_{kl \in E_2} z_{kl} = T_s - 1, \tag{6}$$

$$\sum_{ij \in E(S_1)} y_{ij} \le |S_1| - 1, \qquad S_1 \subset V_1, S_1 \ne \varnothing, V_1, \tag{7}$$

$$\sum_{ij \in E(S_2)} z_{kl} \le |S_2| - 1, \qquad S_2 \subset V_2, S_2 \ne \varnothing, V_2, \tag{8}$$

$$x, y, z \ \text{ binary.} \tag{9}$$

- 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 :

$$\text{maximize} \quad 0 \tag{1}$$

subject to

$$\sum_{k \in V_2} x_{ik} = 1, \qquad \forall i \in V_1, \tag{2}$$

$$\sum_{i \in V_1} x_{ik} = 1, \qquad \forall k \in V_2, \tag{3}$$

$$x \quad \text{binary.} \tag{4}$$

Master solution : Node matching between two graphs

Combinatorial Benders' cuts [8] are recursively added

$$\sum_{i \in C : x^*_{ik} = 0} x_{ik} + \sum_{i \in C : x^*_{ik} = 1} (1 - x_{ik}) \geq 1$$

Slave($\tilde{x}$), a linear system parameterized by $\tilde{x}$:

$$\text{maximize} \quad \sum_{ij \in E_1} y_{ij} \tag{5}$$

subject to

$$\sum_{j \in N(i)} y_{ij} - \sum_{l \in N(k)} z_{kl} = 0, \qquad \forall i \in V_1, \forall k \in V_2, \tag{6}$$

$$\sum_{ij \in E_1} y_{ij} = T_s - 1, \tag{7}$$

$$\sum_{kl \in E_2} z_{kl} = T_s - 1, \tag{8}$$

$$\sum_{ij \in E(S_1)} y_{ij} \leq |S_1| - 1, \qquad S_1 \subset V_1, S_1 \neq \varnothing, V_1, \tag{9}$$

$$\sum_{kl \in E(S_2)} z_{kl} \leq |S_2| - 1, \qquad S_2 \subset V_2, S_2 \neq \varnothing, V_2, \tag{10}$$

$$y, z \quad \text{binary.} \tag{11}$$

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)

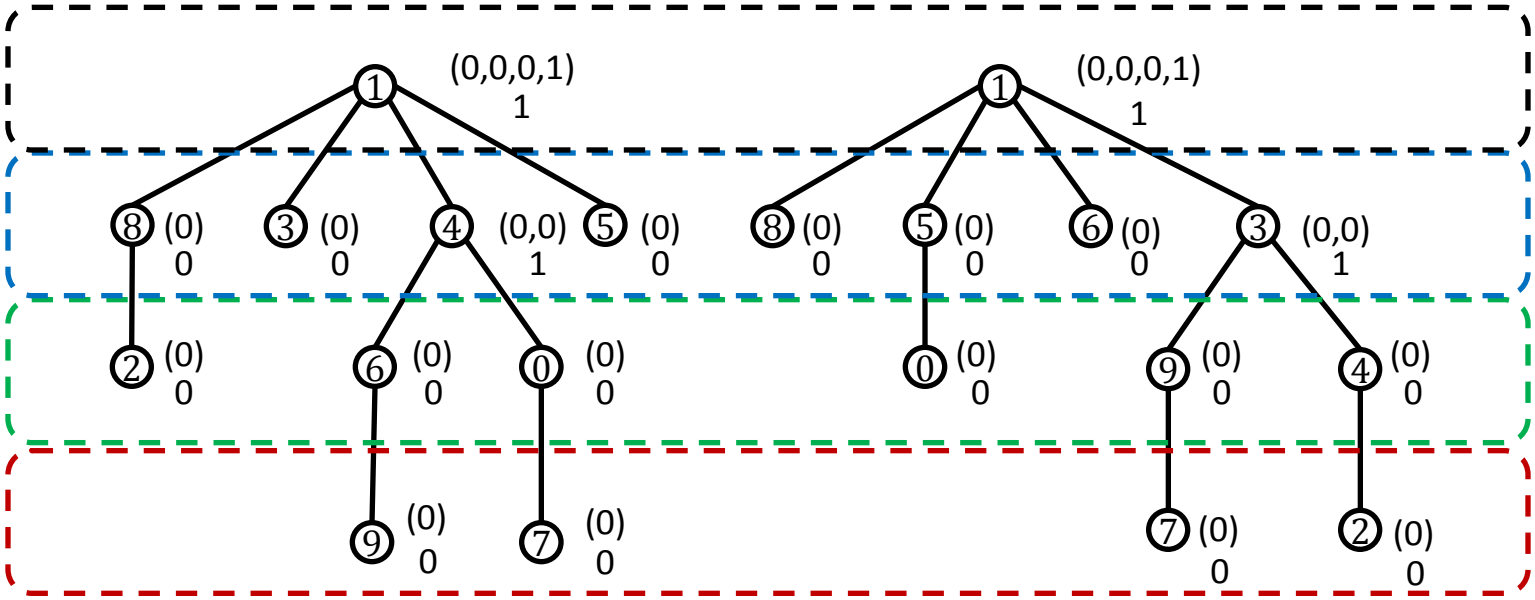


Figure 9 : Checking the tree isomorphism by using the AHU algorithm

Algorithm

Flowchart (in detail)

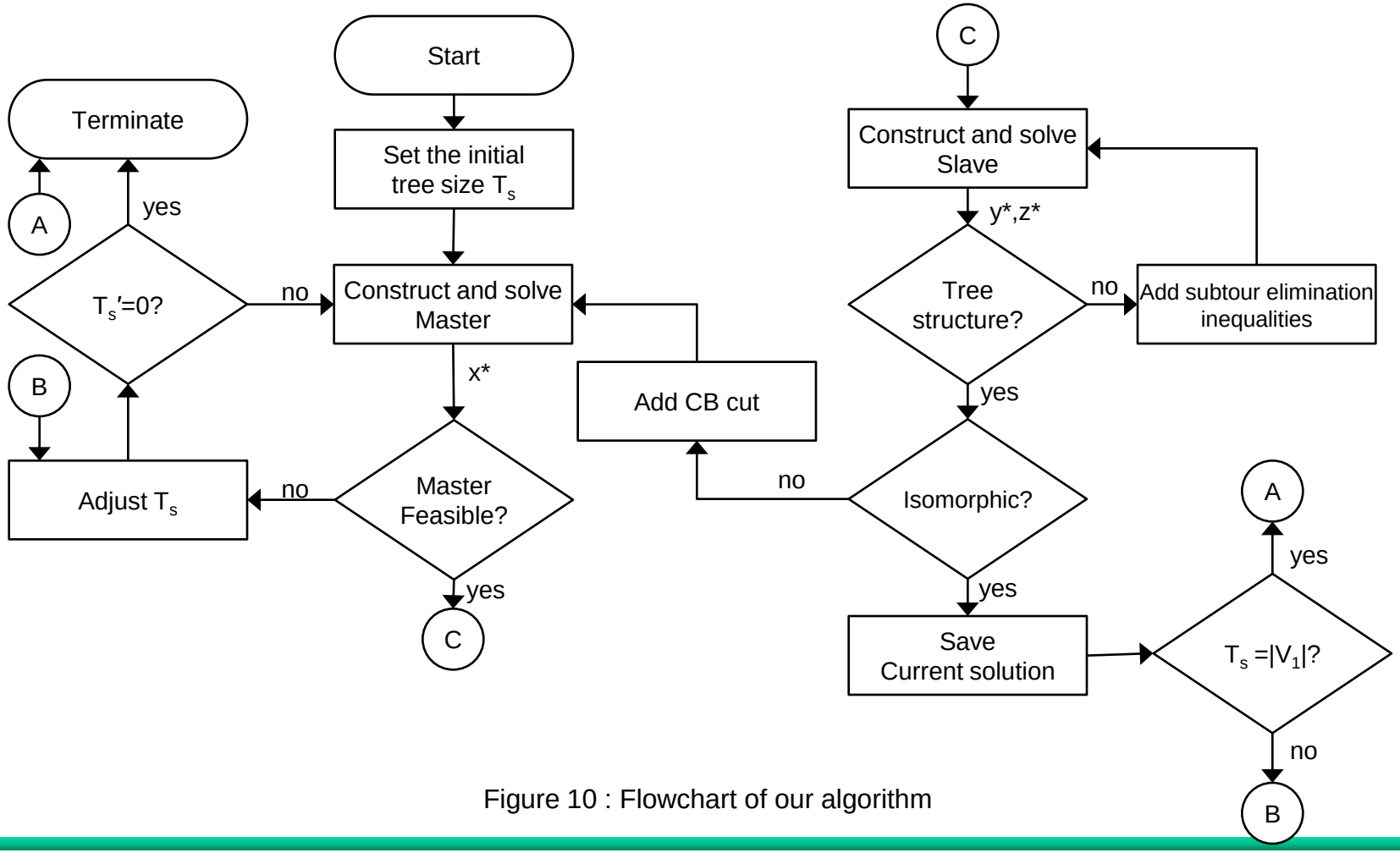


Figure 10 : Flowchart of our algorithm

Contents

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 Erdõs and Rényi model ((n, p) model)
  - Randomly generated trees
  - Scale-free networks
  - Consider the graphs having 10, 20 and 30 nodes

Computational Result

G_1		G_2		Manić et al.'s algorithm		Our algorithm	
$\|V_1\|$	$\|E_1\|$	$\|V_2\|$	$\|E_2\|$	Run time(s)	Solution	Run time(s)	Solution
10	13	10	10	0.2	9	0.2	9
10	11	10	10	0.1	8	0.6	8
10	9	10	11	0.1	8	1.8	8
20	37	20	42	248.2	19	2.3	19
20	36	20	40	195.6	19	2.8	19
20	32	20	41	181.1	19	1.0	19
30	100	30	85	600.0*	28*	14.3	29
30	76	30	74	600.0*	28*	4.44	29
30	91	30	100	600.0*	26*	24.2	29

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

G_1		G_2		Manić et al.'s algorithm		Our algorithm	
$\|V_1\|$	$\|E_1\|$	$\|V_2\|$	$\|E_2\|$	Run time(s)	Solution	Run time(s)	Solution
10	20	10	14	0.1	9	0.1	9
10	15	10	16	1.5	9	0.1	8
10	19	10	20	3.0	9	0.9	8
20	80	20	72	510.2	19	8.6	19
20	84	20	70	600*	16*	2.9	19
20	82	20	80	487.5	19	12.5	19
30	177	30	173	600.0*	28*	261.8	29
30	173	30	171	600.0*	27*	124.8	29
30	179	30	177	600.0*	28*	331.6	29

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

Computational Result

G_1		G_2		Manić et al.'s algorithm		Our algorithm	
$\|V_1\|$	$\|E_1\|$	$\|V_2\|$	$\|E_2\|$	Run time(s)	Solution	Run time(s)	Solution
10	20	10	9	0.1	9	1.2	9
10	14	10	9	0.1	9	0.1	8
10	15	10	9	0.1	9	0.9	8
20	80	20	19	42.7	19	28.2	19
20	72	20	19	85.2	19	21.0	19
20	84	20	19	113.8	19	42.3	19
30	177	30	29	600.0*	28*	151.7	29
30	173	30	29	543.2	29	224.2	29
30	173	30	29	600.0*	27*	351.1	29

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

G_1		G_2		Manić et al.'s algorithm		Our algorithm	
$\|V_1\|$	$\|E_1\|$	$\|V_2\|$	$\|E_2\|$	Run time(s)	Solution	Run time(s)	Solution
10	20	10	9	0.2	9	1.2	9
10	14	10	9	0.1	6	0.1	6
10	15	10	9	0.2	7	0.9	7
20	80	20	19	112.7	15	312.8	15
20	72	20	19	215.1	19	191.0	19
20	84	20	19	343.0	19	312.3	19
30	177	30	29	600.0*	23*	600.0*	23*
30	173	30	29	600.0*	24*	600.0*	23*
30	179	30	29	600.0*	26*	600.0*	26*

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

Computational Result

G_1		G_2		Manić et al.'s algorithm		Our algorithm	
$\|V_1\|$	$\|E_1\|$	$\|V_2\|$	$\|E_2\|$	Run time(s)	Solution	Run time(s)	Solution
10	12	10	10	1.2	8	0.1	8
10	15	10	14	1.4	7	0.2	7
10	17	10	20	2.0	9	0.1	9
20	40	20	42	198.4	15	65.5	15
20	33	20	34	244.1	14	44.2	14
20	29	20	41	87.5	19	8.8	19
30	38	30	44	600.0*	27*	257.5	29
30	55	30	49	600.0*	27*	309.0	29
30	46	30	59	600.0*	20*	527.1	25

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

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

References

[1] V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Massachusetts, 1974.

[2] A. T. Balaban, *Applications of graph theory in chemistry*, Journal of Chemical Information and Computer Sciences, vol. 25, no. 3, pp. 334-343, 1985.

[3] J. R. Banga, *Optimization in Computaional systems biology*, BMC systems Biology, vol. 2, no. 47, 2008.

[4] A. L. Barabási and R. Albert, *Emergence of Scaling in Random Networks*, Science, vol. 286, no. 15, 1999.

[5] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Elsevier Science, New York, 1976.

[6] D. Bertsimas and R. Weismantel, *Optimization over Integers*, Dynamic Ideas, Belmont, 2005.

[7] H. Bunke, *Error Correcting Graph Matching : On the Influence of the Underlying Cost Function*, IEEE Trans. Pattern Analysis and Machine Inteligence, vol. 21, no. 9, pp. 917-922, 1999.

[8] G. Codato and M. Fischetti, *Combinatorial Benders' Cuts for Mixed-Integer Linear Programming*, Operations Research, vol. 54, no. 4, pp. 756-766, 2006.

[9] L. P. Cordella, P. Foggia, C. Sansone, M. Vento, *A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 10, pp. 1367-1372, 2004.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.

[11] E. Gossett, *Discrete Mathematics with Proof*, Pearson Education, Inc, New Jersey, 2003.

[12] J. Hopcroft and R. Tarjan, *Efficient Algorithms for Graph Manipulation*, Communications of the ACM, vol. 16, no. 6, pp. 372-378, 1973.

References

[13] M. Kanehisa and P. Bork, *Bioinformatics in the post-sequence era*, Nature Genetics, vol. 33, no. 3, pp. 305-310, 2003.

[14] B. P. Kelley, R. Sharan, R. M. Karp, T. Sittler, D. E. Root, B. R. Stockwell and T. Ideker, *Conserved Pathways within bacteria and yeast as revealed by global protein network alignment*, Proceedings of the National Academy of Sciences of the United Stated of America, vol. 100, no. 20, pp. 11394-11399, 2003.

[15] W. Kocay and D. L. Kreher, *Graphs, Algorithms, and Optimization*, A CRC Press Company, Boca Raton, 2005.

[16] G. Manić, L. Bahiense and C. Souza, *On the Facial Structure of the Common Edge Subgraph polytope*, the 2008 IEEE Communication Theory Workshop, 2008.

[17] V. Nicholson, C. Tsai, M. Johnson, M. Naim, R.B. King and D.H. Rouvray, *A topological approach to molecular similarity analysis and its application*, Graph theory and topology in chemistry, Elsevier Science, New York, 1987.

[18] P. Raghavan, *Graph Structure of the Web: A survey*, LATIN 2000: Theoretical Informatics (4th Latin American Symposium), 2000.

[19] J. W. Raymond and P. Willett, *Maximum common subgraph isomorphism algorithms for the matching of chemical structures*, Journal of Computer-Aided Molecular Design, vol. 16, pp. 521-533, 2002.

[20] P. Vismara and B. Valery, *Finding Maximum Common Connected Subgraphs Using Clique Detection or Constraint Satisfaction Algorithms*, International Conference on Modeling, Computation and Optimization in Information Systems and Management Sciences 2008, 2008.

[21] H. Whitney, *Congruent Graphs and the Connectivity of Graphs,* American Journal of Mathematics, vol. 54, pp. 150-168, 1932.

Q / A

Supplements

- Revision of Manić et.al's approach for MCST
  - Add inequalities representing the tree structure requirements

$$\sum_{ij \in E_1} \sum_{kl \in E_2} z_{ijkl} = T_s - 1, \qquad (1)$$

$$\sum_{ij \in E(S_1)} \sum_{kl \in E_2} z_{ijkl} \le |S_1| - 1,\ S_1 \subset V_1, S_1 \ne \varnothing, V_1, \qquad (2)$$

$$\sum_{ij \in E_1} \sum_{kl \in E(S_2)} z_{ijkl} \le |S_2| - 1,\ S_2 \subset V_2, S_2 \ne \varnothing, V_2, \qquad (3)$$

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”

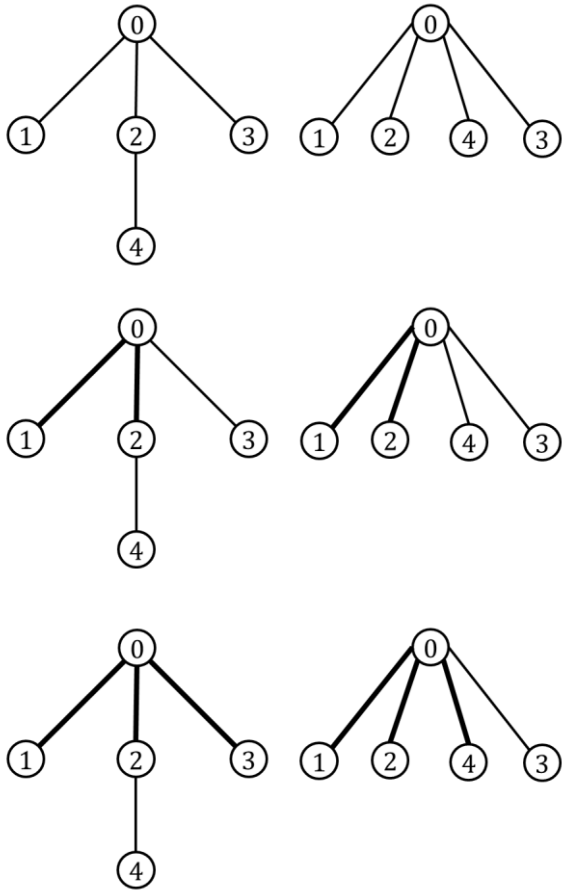
⇩

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 T_s' is adjusted as follows
    - Set $T_s'=\lfloor T_s/2 \rfloor$ at firtst, o/w $T_s'= \lfloor T_s'/2 \rfloor$
    - If the formulation is feasible, then set $T_s=T_s+T_s'$
    - If the formulation is infeasible, then set $T_s=T_s-T_s'$
    - When $T_s'=0$, stop

Supplements

- 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- ⌊5/2⌋=3

When T_s=3,
we can find the common subtree
Set T_s=T_s+T_s'=5+ ⌊2/2⌋=4

When T_s=4,
we can find the common subtree
Since T_s'=⌊1/2⌋=0, the algorithm terminates

Figure 11 : Example of adjusting the tree size