SOFTWARE ARCHITECTURE RECOVERY WITH ERROR TOLERANCE AND PENALTY COST IN GRAPH MINING

Shaheda Akthar[1] , Sk.Md.Rafi [2]

1.Department of Computer science and Engineering,, Sri Mittapalli college of Engineering,A.P. INDIA.
2.Department of Computer science and Engineering,, Sri Mittapalli Institute of Technology for Women,A.P. INDIA

Abstract In the vast literature Software architecture has been recovered through graphs. During Software Architecture recovery generally matching will takes place between Source graph and Query graph. Graph matching is one of the phase in which we will find the same sub graph which is present in source graph and query graph. In most of the papers, it was stated that graph matching is complete. But in reality matching between the graphs is not appropriately complete. We call this an error in matching. In this paper we investigate the applicability of error correcting graph matching algorithm, Graph Isomorphism by Decision Tree to compensate the Software Architecture Recovery.

1. Introduction

Software Architecture Recovery is one of the finest scope to understand the internal logic and components of given software. Several recovery techniques are applied in this context. Among these graph based software architecture recovery [4, 5, 6, 7] is most efficient one. Generally in the recovery process the source graph is divided into sub-graphs of maximum association. To find the association between the graphs, graph mining techniques are used. Once the source graph is decomposed a query graph is generated from the architecture query language. Then an appropriate matching process is used to recover the matched nodes and edges of the graph. Most of the previous papers have stated that graph matching is complete, but in reality matching between the graphs is not appropriately complete. We call this as error in the matching [1,2]. This leads to the error in the recovered software architecture. So in order to compensate the graph error matching, we need to calculate the cost of the matching error. Due to the error in the matching, we need an appropriate error-tolerant, graph matching methods. One way to compensate the error is calculate the graph edit distance [8,9,10]. The edit operations are defined by adding a node, removing the node, adding the edge and removing the edge. In the past, several algorithms are proposed like exact [1, 2, 8, 9] and error tolerant

graphs, graph isomorphism, heuristic search and error correcting graphs. One of the major problems with above algorithms is time complexity, is exponential because the problem is NP complete. Exact matching which requires a strict correspondence among the two objects being matched or their subparts, often fails to provide exploitable results and there is a need to resort to approximate graph matching. Approximate graph matching algorithms allow matching two nodes that violate constraints such as the edge-preservation constraint- exact correspondence of edges or any other characteristics such as node/edge labels, weights etc. Instead a penalty is assigned to those constraints violations depending on the specific problem and desired results. [3]

2. Preliminaries and Notations

2.1 Source graph representation

In this, large software is considered to be the source graph S_G such that $S_G = \{V_G, E_G, x, y\}$ such that V_G and E_G represents the vertices and edges of the source graph. x and y are the label functions which assigns labels to the vertices and edges.

Let a source graph is represented as $S_G = \{V_G, E_G, x, y\}$ with $V_G = \{V_1, V_2, \ldots\ldots V_N\}$. Now the S_G is represented by adjacency matrix $A = \{m_{i,j}\}$,i ,j=1,……,n.

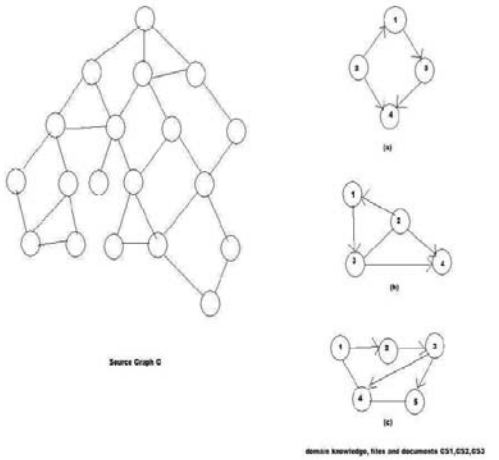

Fig 1 Source Graph with its Sub Graphs

2.2 Permutation Matrix

Now we define the permutation matrix of the given graph by $P=\{p_{i,j}\}$ such that the values $p_{i,j}$ belong to the set {0,1} where i,j=1,…..,n.

The total summation of all values over $\sum_{i=1}^{n} p_{i,j} = 1$ for j=1,…..,n and $\sum_{j=1}^{n} p_{i,j} = 1$ for i=1,….,n.

For a given graph S_G the adjacency matrix A and permutation matrix P are defined such that there exist a matrix $A^2=PAP^T$ where P^T is the transpose the permutation matrix.

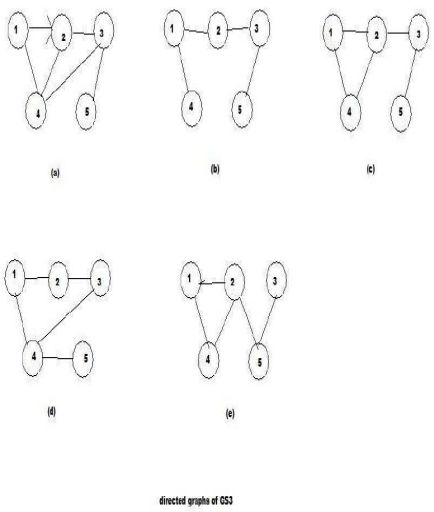

Fig 2 Distorted Graphs of Sub graph belongs to source graph

3. Isomorphic graphs

Two graphs G^1 and G^2 and their corresponding adjacency matrix are $M_1$and M_2. If G^1 is isomorphic to G^2 if the following relation holds for the graph. $M_2=PM_1P^T$.

4. Graph edit operations

For a given graph we have the following edit operations to be performed on the given graph. θ represents the edit function.

1) Adding a node to the corresponding graph
2) Deleting the node
3) Adding the edge
4) Deleting the edge

These four edit operations are most powerful, that translate a graph into corresponding required graph.

5. Edit cost

For a given graph S and corresponding edit operations $D=\{\theta_1,\theta_2\ldots.\theta_n\}$ where n≥1 the edit graph D(S). The total cost to transform the graph from S to D(S) is given by $C(D)=\sum_{i=1}^{k} C(\theta_i)$.

6. Error correcting graph isomorphism

Let the given graphs $G_1$and G_2 such that error correcting graph isomorphism is given by (D,P) where D is the set of edit operations and P is the permutation matrix. The cost of edge correcting graph isomorphism is C(D). The edge correction isomorphism is given by $G_2=PM_{D(G1)}P^1$ where $M_{D(G1)}$ adjacency matrix of the $D(G_1)$.

7. Graph isomorphism by decision tree

7.1 Online and offline

Here we will consider the process to be offline and online. During the offline, source graph adjacency matrix is generated along with several permutations. These permutation matrixes are combined for the decision tree. Now at the time of graph matching, each source graph adjacency matrix are compared to the corresponding AQL query graph. Now in the online phase AQL query graph is transformed in to several permutated sub graphs. These permutated adjacency sub graphs are formed as the decision trees.

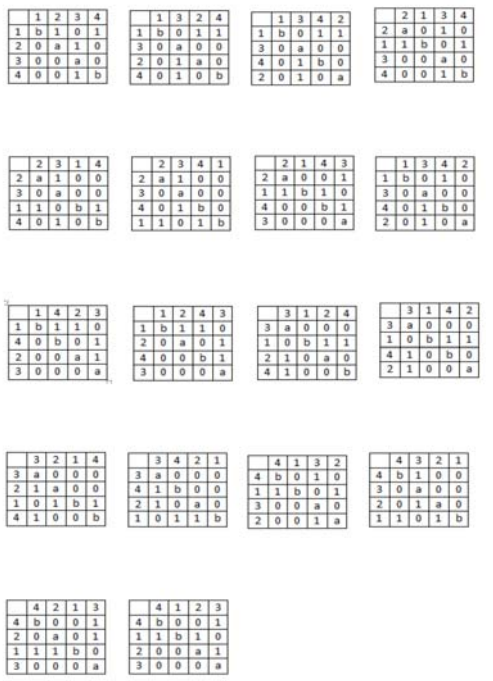

Fig 3 Adjacency Matrix of Sub Graph belongs to Source Graph

7.2 Procedure for mapping process

Let the source graph G_s is divided into a set of sub graphs G_{s1}, G_{s2}……G_{sn} and the graph which is generated from AQL is G_Q. We need to identify the optimal error correcting graph isomorphism (D^i,P^i) between the G_{si} of source graph and G_Q of graph generated from the AQL such that cost $C(D^i)$ is minimal over the sub graphs of source graph G_S. In most of the literatures this problem is solved through A^* algorithm (Kamran Sartipi)[4,5,6,7], it was observed that, A^* suffers from the exponential complexity. So we used the decision tree approach in which, it separates the graph isomorphism from the error correcting process. First take sub graph of source graph which was identified through the domain knowledge separately from the set of distorted graphs. The distance between these distorted graphs are not larger than the threshold value λ. Each of this distorted sub graphs of source graph are separately matched with the graph generated from the AQL query. If the graph distance between G_{si} and G_Q is not larger than λ such that there exist the distorted copy of source sub graph G_{si}, that is isomorphic to the AQL graph G_Q.

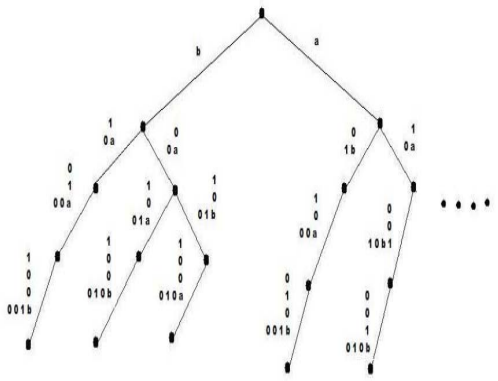

Fig 4 Decision Tree

$D1(G_{si},\lambda)=\{D(G_{si})|D$ is sequence of edit operations with $C(D) \leq \lambda\}$

For each edit operation which are defined from the above sections, are defined a cost of 1. Once the $D(G_{si},\lambda)$ is computed the optimal error correcting isomorphism is determined by testing each query graph G^Q and sub graph of source graph G_{Si}. Algorithm runs in quadratic time complexity, which is a very much improved version than A*, which runs in exponential time. Performance of the matching process has been improved by a factor of $O(Ln^2)$. Same process can be repeated for the query graph G_Q , in which different distortions are generated and compared with the source sub graph G_S such that cost of the edit operations could be less than threshold value λ.

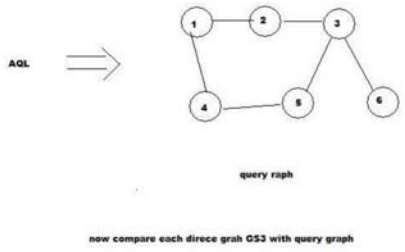

Fig 5 Query Graph Algorithms

1) Let G_{s1},G_{s2},……,G_{sn} are all distorted graphs of sub graph G_S, in turn sub graph of source graph G.
2) Obtain the query graph G_Q from the AQL.
3) Calculate the adjacency and permutation matrix for each distorted graphs, and make the decision tree based on that.
4) Now compute the adjacency matrix and permutation matrix of the query graph.
5) Compare each adjacency and permutation matrix of each graph by the relation $M^Q = M^P = PMP^T$ such that M^Q is the adjacency

matrix of query Graph G^Q and M^P is adjacency matrix of distorted matrix and M is the adjacency matrix of the source graph.

8. Experimental Results

Implementation is done in C++ where a simulation code is developed. Simulation studies shows that our proposed model of graph mining is efficient in time complexity. By due to the decision tree, it suffers from space complexity.

Conclusion In this paper we used an appropriate matching algorithm to compensate the error incurs in matching by Graph Isomorphism and Decision Tree. Experimental results are shown that, our proposed model of matching is more efficient than other matching models.

References

[1] B.T. Messmer, H. Bunke, A new algorithm for error-tolerant sub graph isomorphism detection, IEEE Pattern Anal. Mach. Intel. 20 (1998) 493–504.(6)

[2] B.T.Messmer and H.Bunke. Fast Error- Correcting Graph Isomorphism Based on Model Precompilation IAM- 96-012-Sept 1196 [7]

[3] Error Correcting Graph Matching Application to Software Evolution by Segla Kpodjedo, Filippo Ricca, Philippe Galinier and Giuliano Antoniol

[4] Kamran Sartipi and Kostas Kontogiannis. A user-assisted approach to component clustering. *Accepted for the Journal of Software Maintenance: Research and Practice (JSM),* 2002. [[36]]

[5] Kamran Sartipi and Kostas Kontogiannis. Interactive software architecture recovery: An incremental supervised clustering approach. Technical Report UWE&CE#2002-06, Dept. E&CE, University of Waterloo, Waterloo, Canada, April 2002. [37]

[6] Kamran Sartipi, Kostas Kontogiannis, and Farhad Mavaddat. A pattern matching framework for software architecture recovery and restructuring. In *Proceedings of the IEEE IWPC,* pages 37–47, Limerick, Ireland, June 2000. [38]

[7] Kostas Kontogiannis, R. DeMori, M. Bernstein, M. Galler, and E. Merlo. Pattern matching for design concept localization. In *Proceedings of the Working Conference on Reverse Engineering (WCRE'95),* pages 96–103, 1995. [39]

[8] Sanfeliu, A and Fu, K.S(1983). "a Distance measure between attributed relational graphs for pattern recognition", IEEE Trans. SMC. Vol.13,pp 353-363 [63]

[9] Segla kpodjedo,Filippo Ricca, Philippe Galinicr and Error Correcting Graph Matching Applications to Software Evolution 2008 15th working conference on Reverse Engineering[65]

[10] Tsai, W.H. and Fu, K.S(1979)"Error-correcting isomorphism of attributed relation graph for pattern recognition", IEEE Trans. SMC 9 pp, 757-768[78]

Shaheda Akthar received Bachelor of computer science & Master of Computer Science from Acharya Nagarjuna University, M.S (Software Systems) from BITS, Pilani, Pursuing PhD from Acharya Nagarjuna University. Presently working as Asscociate .Professor in Sri Mittapalli College of engineering, affiliated to J.N.T.U, Kakinada. My area of interest is Software Reliability, Software Architecture Recovery, Network Security, and Software Engineering

Sk.MD.Rafi received B.Tech (comp) from Jawaharlal Nehru Technological University, M.Tech (comp) from Acharya Nagarjuna University. Pursuing PhD from Jawaharlal Nehru Technological University. Presently working as Associate. Professor in Sri Mittapalli Institute of Technology for women, affiliated to J.N.T.U, Kakinada. My area of interest is Software Reliability, Software Architecture Recovery, Network Security, and Software Engineering.