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

5.1. Introduction

Software architecture recovery is the one of the finest of scope to understand the internal logic and components of given software. Several recovery techniques are applied in this context. Among there graph based software architecture recovery [36,37,38,39] 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, a graph mining techniques are used. Once after 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 papers in which stated graph matching is complete, but in reality matching between the graphs is not appropriately complete. We call this as error in the matching[6,7]. 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 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[63,65,78]. 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 to like exact[6,7,63,65]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.

5.2. Preliminaries and Notations

5.2.1 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 \( V_G = \{V_1, V_2, \ldots, V_N\} \). Now the \( S_G \) is represented by adjacency matrix \( A = \{m_{ij}\} \), \( i, j = 1, \ldots, n \).

![Source Graph with its Sub Graphs](image)

5.3 Permutation Matrix

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

The total summation of all values over \( \sum_{i=1}^{n} p_{ij} = 1 \) for \( j = 1, \ldots, n \) and \( \sum_{j=1}^{n} p_{ij} = 1 \) for \( i = 1, \ldots, n \).
For a given graph $S_G$ the adjacency matrix $A$ and permutation matrix $P$ such that there exist a matrix $A^2 = PA^TP$ where $P^T$ is the transpose the permutation matrix.

![Distorted Graphs of Sub graph belongs to source graph](image)

**Fig 5.2** Distorted Graphs of Sub graph belongs to source graph

### 5.4 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.

$M_2 = PM_1P^T$.

### 5.5 Graph edit operations

For a given graph we have following edit operations to be performed on the given graph. $\theta$ 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 in to corresponding requires graph.
5.6 Edit cost

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

5.7 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 $D$ set of edit operations and $P$ permutation matrix. The cost of edge correcting graph isomorphism is $C(D)$. The edge correction isomorphism is given by $G_2=PM_{D(G_1)}P^t$ where $M_{D(G_1)}$ adjacency matrix of the $D(G_1)$.

5.8 Graph isomorphism by decision tree

5.8.1 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 to 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. There permutated adjacency sub graphs are formed as the decision trees.
Procedure for mapping process

Let the source graph $G_s$ is divided in set of sub graphs $G_{s1}, G_{s2}......G_{sn}$ and the graph which is generated from AQL $G_Q$ 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 (Sartipi)[37,37,38,39], it was observed the A* suffers from the exponential complexity. So we used the decision tree approach [ ] in which the it separates the graph isomorphism from the error correcting process. First take sub graph of source graph which was identified through the domain knowledge are separately from the set of distorted graphs. The distance between these distorted graphs are not larger than the threshold value $\lambda$. 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 $\lambda$ such that there exist the distorted copy of source sub graph $G_{si}$ that is isomorphic to the AQL graph $G_Q$.

![Decision Tree](image)

**Fig 5.4 Decision Tree**

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

For each edit operation which are defined from 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 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 $\lambda$.
Algorithm

1) Let the $G_{s1}, G_{s2}, \ldots, G_{sn}$ 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.

5.10. Experimental Results

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