International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
International Conference and Workshop on Communication, Computing and Virtualization (ICWCCV 2015) – www.ijais.org

An Improved Hybrid Approach of Mining Graphs using Dual Active Feature Sample Selection and LTS (Learn-To-Search)

Maya Aikara
Department of Computer Engg,
TCET,

R.R. Sedamkar
Department of Computer Engg,
TCET,

Sheetal Rathi
Department of Computer Engg,
TCET,

ABSTRACT
In software engineering, discriminative sub graphs are used to identify the bug signatures (context of bug). Most of discriminative sub graph mining algorithms estimate the discriminative sub graphs from a positive and negative labelled graph dataset. The labelling is done manually, which is time as well as cost consuming. A hybrid discriminative sub graph mining algorithm using dual active feature sample selection and LTS, which reduces the manual labelling by 60%. But, this hybrid approach does query graph computation without considering the features of the labelled input graph dataset. Even the precision limit is set to 4, which may not be optimal for all type of input dataset. This paper presents an improved hybrid approach, which does a query graph computation considering all graphs in the input dataset. An additional tool is used for input pre-processing method. The average precision limit is determined so as to achieve maximum recall for any type of input dataset. The experiments and results shows that the improved hybrid approach can achieve an average recall of 66.67% when the precision limit is set to 3, whereas the earlier hybrid approach attained an average recall of 33.33% when precision limit was set to 4.

Keywords
Graph Mining, Discriminative sub graph mining, Bug Signatures

1. INTRODUCTION
The software engineering (SE) data are generally represented as graphs for better understanding of its attributes and relationships [1]. These graph data are usually static or dynamic call graphs, or program dependence graph. Mining of the SE graph data is done for debugging or bug detection [2]. Certain bugs cannot be identified when scanned individually, but can be identified when executed one after the other. The context of such bug is called bug signatures [3]. There are various frequent sub graph mining algorithm [4,5] which can be used to find bug signatures. The buggy location in the program flows may not be always frequent and so frequent sub graph mining may not be efficient. An alternative option is the discriminative sub graph mining algorithms, which identifies the discriminative sub graphs using a set of positive and negative labelled graphs. This discriminative sub graph represents the bug signatures in the program flow graphs.

There are various discriminative sub graph mining algorithms which can be used for identifying bug signature from program flow graphs. The subdueCL[6] is a graph-based concept learning method which guarantees a discriminative sub graph pattern for each positive graph, but lags efficiency since sub graph frequency is estimated using sub graph isomorphism. The Leap [7] discriminative sub graph mining algorithm correlates structural similarity and significance similarity, but for optimality guarantee, the leap length of 3/4 should be set to 0. It is not efficient for tough dataset. The CORK algorithm [8] reduces the problem of redundancy and significance; still it is inefficient as discriminative sub graph with different discriminative power may have same number of correspondences. The GraphSig [9]tackles the problem of huge search space and produces higher accuracy than LEAP, but identified discriminative sub graph may not be optimal because there might be structural loss of information while sliding the window. In COM [10] heuristic pattern exploration order and co-occurrences can improve runtime efficiency of mining discriminative patterns. It uses frequency as a measure to estimate the discriminative score, which is not always optimal. The GAIA [11] employs sub graph encoding approach to support an arbitrary sub graph pattern exploration order, but posses the risk of missing optimal solution due to the fitness score and random pattern exploration.

The LTS (Learn-to-Search)[12] integrates greedy approach or a branch and bound approach, to resolve the problem of huge search space and reach optimization. The input to LTS is set of positive and negative labelled graph dataset. The method first follows a greedy approach (fast probe) to find the near-optimal solution. Using this near-optimal solution a prediction tree and score record is built to estimate the upper bound of patterns. Finally the LTS algorithm is used to compare the upper bound and find the most discriminative sub graph for very positive graph in the input dataset. The LTS methodology has an improved runtime compared to Leap, GAIA. Moreover, it has a risk-free approach of missing the optimal solution as it employs a multi-lineage pattern exploration.

Most of the discriminative sub graph mining algorithm has the input as a labelled set of positive and negative graph dataset. The labelling of the graph is generally done manually, which is time and cost consuming approach. The dual active feature sample selection [13] is a discriminative

sub graph mining algorithm used for classification. It reduces the manual labelling of the graph input dataset by employing active learning along with feature selection. The simultaneous approach of active learning and feature selection helps to identify the query graph for the set of unlabelled input graph dataset. The query graph is used for classification. The query graph poses a set of discriminative sub graphs. The dual active feature sample selection use a single-lineage pattern exploration, has a risk of missing the optimal solution.

The hybrid approach of dual active feature sample selection and LTS[14] is preferred as it reduces manual labelling without the risk of missing the optimal solution. The LTS has an improved runtime compared to state-of-art discriminative sub graph mining algorithms. The hybrid approach uses the discrimination score estimation of dual active feature sample selection. Then, it replaces the recursive feature selection algorithm, gSpan [15] with the LTS algorithm. Finally a query graph is selected from the set of unlabelled graph dataset. The query graph posses a set of discriminative sub graphs, which are used to identify bug signatures. The hybrid approach reduces manual labelling by 60%. The precision and recall is improved by 33.33%. The hybrid approach has following drawbacks. Firstly, the precision is measured in the scale of 4, which means only precision limit is set to 4. This limitation has a scope of improvement as the precision limit may vary based on the structure of input graphs. Secondly, the query graph is selected only from the set of unlabelled graph. The features of labelled graph are ignored. Thirdly, tool was not used for conversion of execution graph of inputs to adjacency matrix. In this paper, an improved hybrid approach is introduced to avoid these limitations.

The remaining part of paper is organized as follows. Section II gives the introduction to Dual active feature sample selection and LTS method. Section III describes the hybrid approach of dual active feature sample selection and LTS in detail. Section IV elaborates the modification done on the hybrid approach to achieve improved results. Section V describes the experiment conducted and the results obtained. Section VI, defines the future scope. Section VII concludes the paper.

2. RELATED THEORY

2.1 Dual active feature sample selection

Most of conventional approaches to find discriminative sub graph features mine under supervised setting. They assume that labelled graphs are available in the real world domain. This is not always true. The labelling of graph data is time consuming and an expensive task. To reduce the manual labelling, active query learning is employed, which selects a query graph for labelling. In graph database, both active learning and feature selection technique are correlated. Therefore, active learning problem and feature selection problem in graph data can be considered simultaneously as shown in Figure 1. The combined approach is called dual active feature and sample selection[13].

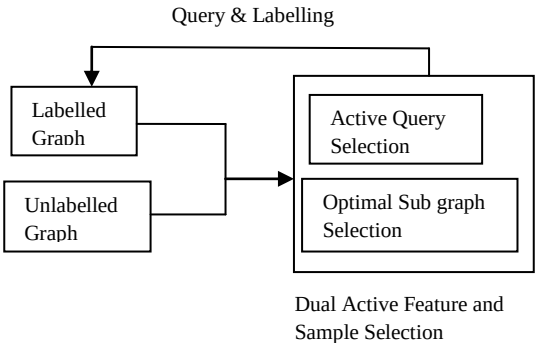

Fig 1: Dual Active Feature & Sample Selection

The combined approach minimizes the manual labelling by selecting a query graph for label. This query graph is representative as well as informative. The query graph is considered as the discriminative sub graph while mining. The feature mining is done with gSpan algorithm [15]. The gSpan algorithm uses single lineage pattern exploration, which poses the risk of losing the optimal solution. The branch and bound algorithm is used to prune the search space. The dual active feature and sample selection reduces the manual labelling drastically, thus leading to better graph classification. It also provides the optimal feature set and query graph which is useful for discriminative sub graph mining. As it uses single lineage exploration, it might lose the optimal feature once pruned.

2.2 LTS(Learn-to-Search):Discriminative sub graph mining algorithm

The discriminative sub graph characterizes complex graphs, construct graph classifiers or create graph indices. The discriminative score cannot be measured using sub graph frequency as the discriminative sub graph are neither monotonic nor anti-monotonic. Thus, the branch-and-bound algorithms are not very efficient for mining discriminative sub graphs. The LTS[12] uses the search history for better computation of upper bounds of discriminative score. For discriminative sub graph mining, the search space is very large. To optimize the search problem there are two ways (i) greedy approach (only the sub graph with higher score are considered and others pruned. It is faster approach) (ii) branch-and-bound approach (uses the upper bound for pruning the search space). The LTS combines the two approaches. It first uses a greedy method(fast probe algorithm[12]) to find the near-optimal solution and builds up the search history(prediction tree and table). Later it estimates the upper bound using the search history generated by fast probe and finds the discriminative sub graphs. It takes a labelled set of positive and negative graph set as input and the optimal discriminative pattern for each graph in positive set. LTS employs multi-lineage pattern exploration which provides lesser risk of missing the optimal solution. As LTS uses fast probe (aggressive pruning over multi-lineage pattern exploration), it runtime is improved or similar to the state-of-art discriminative sub graph algorithms. It has the drawback that all the input graph dataset needs to be manual labelled before computation of discriminative sub graph.

3. A HYBRID APPROACH OF DUAL ACTIVE FEATURE SAMPLE SELECTION AND LTS

The hybrid approach [14] is combining the existing methods of dual active feature sample selection and LTS(Learn to Search). The dual active feature sample selection has a good discriminative score calculation and simultaneous computation of labelled and unlabelled features, but the feature selection is done with gSpan algorithm which employs single lineage pattern exploration. The pruning on single lineage pattern exploration posses the risk of missing optimal patterns. The LTS is a risk free approach of missing the optimal patterns due to multi-lineage pattern exploration, but the discriminative score estimation is based on frequency of pattern in dataset. Thus, the hybrid approach combines the advantages of the two methods and overcomes their drawbacks as shown below.

Dual Active Feature and Sample Selection	**LTS(Learn to Search): Discriminative subgraph mining algorithm**
Advantages:- ➢ Reduces the labelling cost of the graph data and selects query graph for labelling ➢ Estimates the usefulness of query graph and set of subgraph features simultaneously ➢ Accurate compared to alternate approaches ➢ Better runtime efficiency as reduction in manual labelling work of graph data	Advantages:- ➢ Uses multi lineage pattern exploration with aggressive pruning to achieve faster and better optimal solution. ➢ Integrates greedy approach(for local optima) and branch-n-bound approach(prune search space) ➢ LTS efficient runtime compared to Leap, GAIA ➢ LTS is same or more accurate compared to state-art-algorithms.
Disadvantages:- ➢ Risk of missing the optimal solution because pruning is carried on single-lineage subgraph pattern exploration	Disadvantages:- ➢ Labelled input of graph data set to find discriminative subgraph, which is mostly done manually. ➢ Discriminative score is frequency-based computation.

Hybrid Approach
Advantages:- ➢ Reduces the labelling cost of the graph data and selects query graph for labelling ➢ Provides a risk-free optimal solution ➢ Estimates the usefulness of query graph and set of subgraph features simultaneously ➢ Integrates greedy approach(for local optima) and branch-n-bound approach(prune search space) ➢ Better runtime efficiency as manual labelling is reduced & aggressive pruning.
Disadvantages:- ➢ The query graph is estimated only from the sub graph features of unlabelled graph datatset ➢ The optimal precision limit varies with the structure of the input graph datatset ➢ The conversion of execution graph of input to the corresponding adjacency matrix is done manually.

Fig 2: Derivation of hybrid approach

4. IMPROVED HYBRID APPROACH

The Hybrid approach of dual active feature sample selection and LTS to identify bug signatures from the program flow graphs has a scope of improvement in the following areas:-

4.1 Input pre-processing

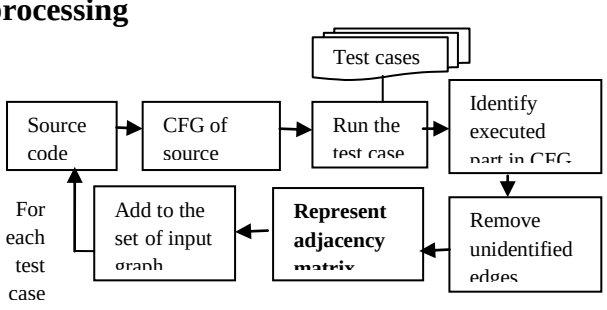

Fig 3: Input pre-processing

To identify bug signatures from the program source code, the input to the proposed approach needs to be pre-processed as shown in the Figure 3. The input to our methodology is a set of labelled and unlabelled graph of execution traces of a source program. The flow from source program and test cases to the set of input graph is explained as follows. The source program is written and complied. The control flow graph(CFG) of the program source code is generated. The nodes represent a single statement and edges represent the control flow between these nodes. The control flow graphs are generated with the help of tool called control flow graph tool(eclipse plugin)[16]. Then, one test case is applied to get the execution traces. This execution trace is then used to identify the part of control flow graph(CFG) of program which was executed. For the statements which were not executed, the control flow edge and nodes are removed from the source code CFG. The code coverage during execution is identified with Code Cover tool(eclipse pug in)[17].The adjacency matrix of the modified CFG is obtained and this matrix is added to the set of input graphs. The hybrid approach algorithm requires adjacency matrix representation of the input for further processing. Similar, method is used to acquire input graph for the remaining test cases of the program source code.

The adjacency matrix in the hybrid approach is obtained manually. A tool named SocNetV (open source tool)[18] is a social networking analysis tool. It can import a GraphViz .dot file or an .xml file and convert into the corresponding adjacency matrix. The improved hybrid approach has its input pre-processed along with the SocNetV tool, in addition to the existing tools. The addition of this tool helps in faster pre-processing of the input.

4.2 Estimation of query graph.

The query graph posses the set of discriminative sub graphs, which are the bug signatures in the program flow graphs. The query graph therefore needs to be informative as well as representative in the cluster of graphs. The formula for query graph estimation in the hybrid approach is as follows:

$$G_s^* = \max_{Gs \in Da} \sum_{g i \in Ti} h(g_i, M) \quad \ldots.1$$

where $h(g_i, M)$ is the discrimination score function

g_i is the each graph in the each feature list T_i

D_a is the total number of unlabelled graph in the dataset

For each unlabelled graph, a set of k discriminative sub graphs(g) having the higher scores are identified. Their scores are stored in T. The query graph is that unlabelled graph, which is having maximum score of sum of the scores of their discriminative sub graphs.

In the hybrid approach, the query graph is selected only from the set of unlabelled graphs. The features of labelled graph are ignored. This leads to a risk of missing the optimal features which might be a bug signature. Therefore, the improved hybrid approach changes the query graph formula to:

$$G_S^* = \max_{Gs \in D} \sum_{gi \in Ti} h(g_i, M) \quad \ldots.2$$

The D_a in the query graph formula is replaced with D where D represent all graph in the datatset(labelled and unlabelled)

4.3 Precision limit computation

In the hybrid approach, the precision limit was set to 4 and precision was measured in the scale of 4. The average recall achieved is 33.33%. When the precision limit is increased(precision value is high), the recall lowers. This happens because as more number of sub graphs are identified, the optimal features are averaged with non-optimal features. When the precision limit is very lowered(precision value is very low), the optimal feature may be missed and the recall gets lowered. Therefore, the optimal precision value needs to be estimated. The precision limit is generally based on the structure of the optimal precision limit is different for similar structured input graph dataset, differently structured input graph dataset or both. In the improved hybrid approach, the optimal precision is estimated for each type of input graph dataset and an average is computed. This computed average precision limit will be useful to find nearest optimal features for any type of input graph dataset.

5. EXPERIMENTS AND RESULTS

The experimentation is carried out in Java on 1.7 Ghz i5 core processor, 3.86 memory with Windows 7 64bit operating system. Eclipse Kepler environment is used for experimentation. The inputs are three programs in java

- Program 1: a program to find whether the number is prime or not
- Program 2: given a number to display the month of a year
- Program 3: to carry out insertion sort

The Program 1 is a simple java program with no methods. It has a simple logic to find whether the number is prime or not. The Program 2 is a java program having switch case to identify the month of the year. The Program 3 does insertion sorting for given array of elements. This program has methods to carry out sorting. The bugs are manually inserted in the program such that the logic of the program changes and output might be errorneous for certain inputs. This means that the program will be executed , but the expected output may not be obtained. For each program a set of 10 test cases are selected. The test cases are choosen based the similarity in the structure of the execution graphs. The Program 2 test cases are choosen such that the execution graph of each case are different in structure. The test cases for Program 3 gives the similar execution graphs for correct and faulty outputs. Whereas the test cases for Program 1 provides both similar and dissimilar execution graphs .

Before applying the hybrid approach, the input needs to be pre-processed. In the improved hybrid approach, one more tool is added called SocNetV(social networking analysis tool)[18]. It is an open source tool generally used to anaylise the networks. It also posses a property of importing a GraphViz .dot file or .xml file and exporting it to the corresponding adjacency matrix representation. A snapshot of this tool is shown in the Figure 4.

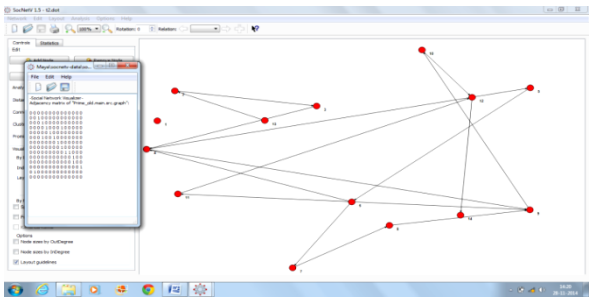

Fig 4: **Snapshot of the adjacency matrix for the control flow graph of Program 1 using SocNetV tool**

The performance metrics used to measure and analysis results are precision and recall. The runtime of the improved hybrid approach is almost same as the original hybrid approach since, the percentage of manual labelling the input is not changed. Only 40 % of the input graph dataset is labelled in the improved hybrid approach. The change in the query graph formula does not much tamper the execution runtime of the hybrid approach. Hence, only precision and recall are performance metrics considered for comparision.

The precision is estimated as the number of discriminative sub graph possessed by the query graph[14] and recall is the computed as the percentage of difference between the positive ratio and negative ratio[14]. The precision and recall are compared program-wise for easy and better understanding of the behaviour in terms of the structure of the input graph dataset. The precision and recall results for the program 1(prime numbers) is as shown in Table 1 and Figure 5.

Table 1. Results of precision and recall for program 1

Sr.No.	Precision	Query graph no.	Recall
1	1	8	50
2	2	8	50
3	3	8	50
4	4	8	37.5
5	5	8	20
6	6	8	16.66667
7	7	8	14.28571
8	8	8	12.5
9	9	8	11.11111
10	10	8	10
11	11	8	9.090909

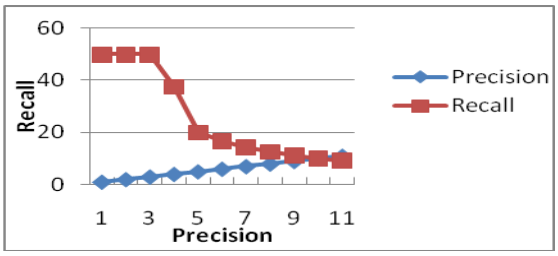

Fig 5: Analysis for program 1

In the program 1, it is visible that the query graph is not changed for any the precision. It can also be seen that the recall is highest for the first three precision value. The optimal precision value should be considered as 3, since it as maximum number of discriminative sub graphs(3) when compared with the precision value 1. More the optimal discriminative sub graphs identified, the better is the bug signature estimation. The maximum recall obtained is 50% with the maximum precision limit as 3.

Table 2. Precision and Recall results for Program 2

Sr.No.	Precision	Query graph no.	Recall
1	1	4	50
2	2	4	16.66667
3	3	4	-16.6667
4	4	4	0
5	5	4	10
6	6	7	16.66667
7	7	7	14.28571
8	8	7	11.11111
9	9	7	11.11111
10	10	7	10
11	11	7	9.090909
12	12	7	8.333333
13	13	7	7.692308
14	14	7	7.142857
15	15	7	6.666667

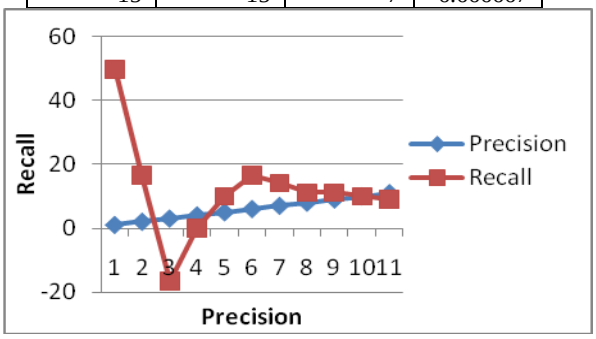

Fig 6: Analysis for program 2

In the program 2, the query changes for some of the precision values, but the discriminative sub graphs obtained is almost similar. This happens because each of the input graph in the dataset is differently structured. The optimal precision for such program is 1 as the recall is highest for it. The recall obtained is only 50 % as there are two positively labelled graph and two negatively labelled graph in the dataset and every graph has a different structure in the input dataset.

Table 3. Precision and Recall results for Program 3

Sr.No.	Precision	Query graph no.	Recall
1	1	7	100
2	2	7	100
3	3	7	100
4	4	7	100
5	5	7	100
6	6	7	66.66667
7	7	7	57.14286
8	8	7	37.5
9	9	7	33.33333
10	10	7	30
11	11	7	27.27273
12	12	7	25

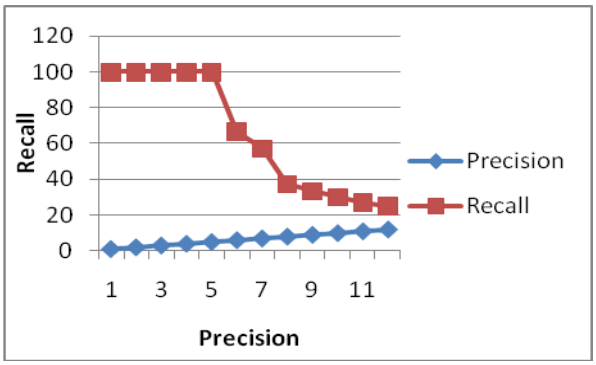

Fig 7: Analysis for program 3

In the program 3, the query graph remains the same throughout. As the precision is changing, recall also fluctuates. The optimal precision is to be considered as 5, since it is lowest precision with high recall value. The lower is the precision, more is the discriminative sub graphs identified and more is the bug signatures found. The recall value is 100% because all the input graphs in the dataset are similarly structured.

Table 4. The overall Precision and Recall results for three programs

	precision	Max recall %
Program 1	3	50
Program 2	1	50
Program 3	5	100

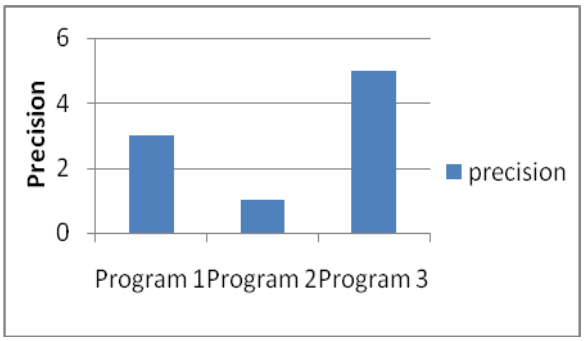

Fig 8: Analysis for precision

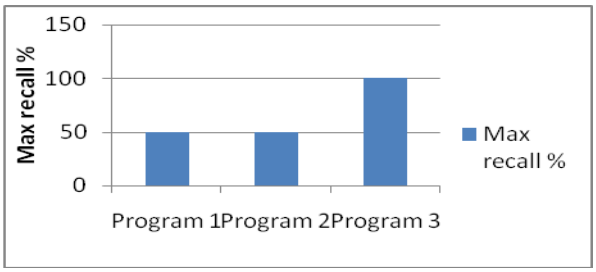

Fig 9: Analysis for recall

The overall results and analysis of recall and precision is shown above. The average precision is 3 and average recall is 66.67% irrespective of the structure of the input program. This precision limit gives the nearest optimal solution to identify the bug signatures. This result overpowers the average recall of earlier hybrid approach[14] which attained an average recall of 33.33% for a precision limit of 4.

6. CONCLUSION AND FUTURE SCOPE

This paper presents an improved hybrid approach of mining graphs using the two concepts, they are dual active feature sample selection and LTS(Learn-to-Search). The existing hybrid approach had reduced 60% of manual labelling and removed the risk of missing the optimal solution, but had some drawbacks. In the input pre-processing of the hybrid approach, the conversion from execution graphs to the adjacency matrix representation was done manually. The query graph was estimated only on the basis of unlabelled input graphs, labelled graphs were ignored. The precision limit considered was not the optimal one.

To overcome these drawbacks, the improved hybrid approach is designed which added a tool SocNetV for the conversion of execution graphs to the adjacency matrix. The query graph formula is modified so as to consider labelled graphs along with unlabelled graphs. The varied precision limits were compared to identify the average optimal precision 3 for any kind of input graph dataset. This precision limit helps to reach the maximum recall of 66.67%. The improved hybrid approach can help the programmer for easy and better identification of discriminative sub graphs, which in return will help in identifying the buggy locations in the program source code.

As the average precision limit helps to find the nearest-optimal solution of discriminative sub graph with maximum recall and minimum runtime, still there is a scope to reach the best optimal solution. The precision limit can be automated depending upon the structure of the input graph dataset.

7. REFERENCES

[1] Wang, Haixun. *Managing and mining graph data*. Edited by Charu C. Aggarwal. Vol. 40. New York: Springer, 2010.

[2] Xie, Tao, Suresh Thummalapenta, David Lo, and Chao Liu. "Data mining for software engineering." *Computer* 42, no. 8 (2009): 55-62.

[3] Cheng, Hong, David Lo, Yang Zhou, Xiaoyin Wang, and Xifeng Yan. "Identifying bug signatures using discriminative graph mining." In *Proceedings of the eighteenth international symposium on Software testing and analysis*, pp. 141-152. ACM, 2009.

[4] Jiang, Chuntao, Frans Coenen, and Michele Zito. "A survey of frequent subgraph mining algorithms." *The Knowledge Engineering Review* 28, no. 01 (2013): 75-105.

[5] Eichinger, Frank, Klemens Böhm, and Matthias Huber. "Mining edge-weighted call graphs to localise software bugs." In *Machine Learning and Knowledge Discovery in Databases*, pp. 333-348. Springer Berlin Heidelberg, 2008.

[6] Gonzalez, Jesus A., Lawrence B. Holder, and Diane J. Cook. "Graph-based relational concept learning." (2002).

[7] Yan, Xifeng, Hong Cheng, Jiawei Han, and Philip S. Yu. "Mining significant graph patterns by leap search." In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp.433-444.ACM,2008.

[8] Thoma, Marisa, Hong Cheng, Arthur Gretton, Jiawei Han, Hans-Peter Kriegel, Alexander J. Smola, Le Song, S. Yu Philip, Xifeng Yan, and Karsten M. Borgwardt. "Near-optimal Supervised Feature Selection among Frequent Subgraphs." In *SDM*, pp. 1076-1087. 2009.

[9] Ranu, Sayan, and Ambuj K. Singh. "Graphsig: A scalable approach to mining significant subgraphs in large graph databases." In *Data Engineering, 2009.ICDE'09 IEEE 25th International Conference on*,pp.844-855.IEEE,2009.

[10] Jin, Ning, Calvin Young, and Wei Wang. "Graph classification based on pattern co-occurrence." In *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 573-582. ACM, 2009.

[11] Jin, Ning, Calvin Young, and Wei Wang. "GAIA: graph classification using evolutionary computation." In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 879-890. ACM, 2010.

[12] Jin, Ning, and Wei Wang. "LTS: Discriminative subgraph mining by learning from search history." In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pp. 207-218. IEEE, 2011.

[13] Kong, Xiangnan, Wei Fan, and Philip S. Yu. "Dual active feature and sample selection for graph classification." In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 654-662. ACM, 2011.

[14] Aikara, Maya, R. R. Sedamkar, and Sheetal Rathi. "A Novel Approach using Dual Active Feature Sample Selection and LTS (Learn to Search)."*International Journal of Computer Applications* 101, no. 13 (2014): 31-36.

[15] Yan, Xifeng, and Jiawei Han. "gSpan: Graph-based substructure pattern mining." In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pp. 721-724. IEEE, 2002.

[16] http://eclipsefcg.sourceforge.net/ Tutorial for CFG generator, Eclipse plugin

[17] http://codecover.org/documentation/references/eclManual.html Tutorial for Code Cover tool, Eclipse plugin

[18] http://socnetv.sourceforge.net/docs/manual.html Tutorial for SocNetV (Social Network Visualizer) tool.