Bachelor Thesis
June, 2022

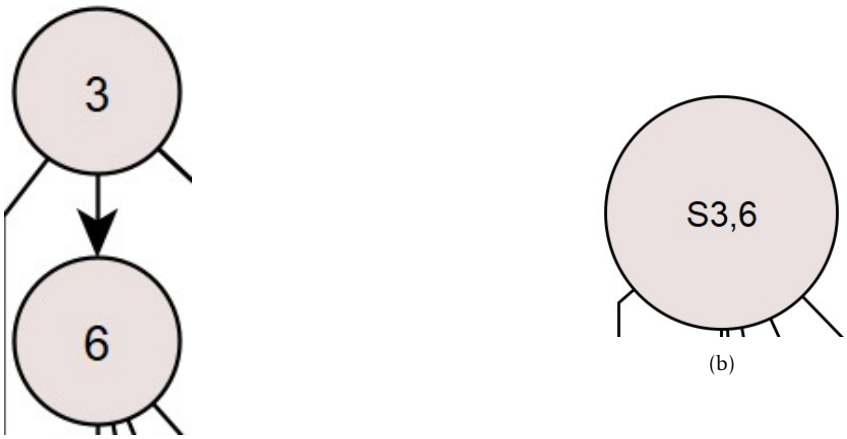
Απόστολος Μαυρογιαννάκης
3799

1 Introduction

We take into account a graph with many nodes and edges. Such a graph has the drawback of being challenging to make visible. As a result, we used a number of techniques to try and summarize this graph and make it visible to the naked eyes. Furthermore, we found a way to construct a relationship between two nodes that are not connected by any edge, neither directly nor indirectly.

2 Topology Neighborhoods

Here we introduce super nodes. A super node contains all the consecutive nodes that might appear in each path, plus it has twice the size of a regular node.



(a) (b)

Figure 1: Combination of consecutive nodes into a super node

Super node's label demonstrates the nodes in the summary. For instance, in the figure above (Picture (b)), the super node contains nodes 3 and 6. Additionally, Super node's coordinates are calculated based on the last node that has a cross or jumping cross edge.

In the first step of the pseudo code (Algorithm 1), Topology Neighborhoods are defined. The algorithm iterates through the nodes of the graph and searches for nodes that belong in the same path. If those nodes are consecutive they are added in the same Summary, else a new summary is created for each of them.

Algorithm 1 Finding Topology Neighborhoods

Input a path decomposition Sp = (P1,...,Pk) of a DAG G; a path based hierarchical drawing Γ_0 according to Variant 0

Output a path based hierarchical drawing Γ_s according to Summaries

```
// 1.Define Summaries
for every Path P in G do
    for each pair (u,v) of vertices in P do
        Let (X(v),Y(v)),(X(u),Y(u)) be the x-y coordinates of vertices v,u respectively
        if |Y(v) − Y(u)| == 1 then
            Add u,v in the same summary
            if u have a cross edge and Y(u) ¿ Y(v) then
                Update Summary's y-coordinate
            end if
        else
            Create new Summary
        end if
    end for
end for

// 2. Fix edges and bends
for each edge e=(u,v) in Γ0 do
    Let Su, Sv be the Topology Summaries of u,v respectively
    Add in Su's targets the summary Sv
    Add in Sv's sources the summary Su
    if X(Su) ≠ X(Sv) then
        Add a bend since this is a crossing edge
    else if |Y(Sv) − Y(Su)| > 1 & there are internal Summaries then
        Add two bends since this is a path transitive edge
    end if
end for
```

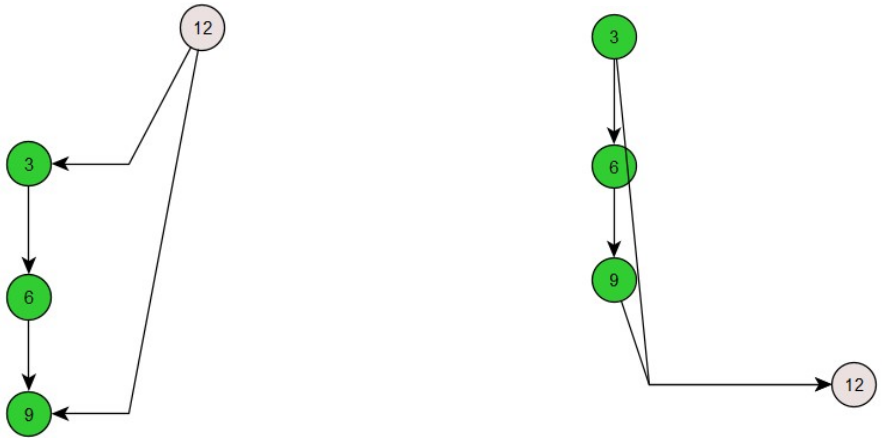
The second step of the pseudo code corresponds to fixing the edges, meaning adding the necessary bends so there is no collision between nodes and edges. This is done by checking for internal Topology Summaries. To achieve this step, a **Summary Channel** is introduced. A summary channel is the same as a path, but it contains Topology Neighborhood Summaries instead of nodes. Each time a Topology Neighborhood Summary is created, that summary is inserted into a Summary Channel based on its X-coordinate.

3 Semantic Summary

This technique is based on cross and jumping cross edges. More specifically, a semantic summary contains two nodes that appear in the same path and have a common source or target, for simplicity, we will refer to

these nodes as boundaries since they are the first and last node of the summary, and we will refer to the common source/target as the **core** node of the semantic summary. A semantic summary also contains the internal nodes of these boundaries.

For instance, in Figure 2(a) nodes 3,6 and 9 create a semantic summary because nodes 3 and 9 (boundaries) are connected with node 12 (common source). Additionally, in Figure 2(b) nodes 3,6, and 9 create a semantic summary since nodes 3 and 9 reach node 12. In both examples, the core node of both Semantic Summaries is node 12.



(a) Common Source Summary

(b) Common Target Summary

Figure 2: Examples of semantic Summaries

Furthermore, a Semantic Summary Channel is introduced. A Semantic Summary Channel is an ordered list containing all the Semantic Summaries existing in a path. The summaries are sorted based on the Y-coordinate of their first node. Semantic Summary Channels are very important in finding overlapping summaries mentioned below.

Algorithm 2 Find Semantic Summaries

Input a path hierarchical drawing Γ_0 according to Variant 0, a DAG G
Output a path based hierarchical drawing Γ_s according to Semantic Summaries

```
// Defining Semantic Summaries
for every vertex u in G do
    Let S be a Map < K, V > where K is the x-coordinates of nodes that reach to u (sources) and V is a
Semantic Summary containing all those nodes
    Let T be a Map < K, V > where K is the x-coordinates of nodes that u points to (targets) and V is a
Semantic Summary containing all those nodes
    for every vertex s that reaches to u (source) do
        Add s in Summary S[X(s)]
    end for
    for every vertex t that u points to (target) do
        Add t in Summary T[X(t)]
    end for
    for every Summary sum in S,T do
        if sum contains more than one nodes then
            Find internal nodes between sum.FirstNode and sum.LastNode
            Add this sum to the Semantic Path
        end if
    end for
end for
```

Nodes that belong to a semantic summary are represented with a different colour. The colour is assigned based on the **number of semantic summaries each node belongs to**. In more details, table below shows which colour corresponds to the number of summaries a node might belong to.

Summaries	Color
0	Gray (default)
1	Green
2-3	Red
4-5	Brown
6-7	Yellow
8-9	Orange
10-11	Blue
12+	Purple

For example, in Figure 2(a) nodes 3,6,9 are coloured green because they belong to one semantic summary.

4 Overlapping Summaries

Overlapping summaries are the pairs of semantic summaries that have common nodes. For instance, Figure 3 demonstrates two semantic summaries – Summary 1 with nodes 3,6,9,2 and Summary 2 with nodes 6,9,2,4,0 –. The core nodes of both summaries are not presented here, since they are not used to make our point. These two summaries are overlapping since they have three common nodes — nodes 6,9,2—.

Figure 3: Caption

Overlapping Summaries can be used to filter out Semantic Summaries using a threshold. The algorithm loops through all of the Semantic Summaries in the graph and finds all of the colliding summaries for each summary. Moreover, the algorithm introduces two options on how to filter the outcome. The first option (relaxed search) prints the pairs of Semantic Summaries that at least one of them satisfies the threshold given by the user. Whereas, the other option (rigorous search) searches for pairs of Semantic Summaries that both satisfy the threshold given by the user.

Overlapping Summaries can reveal some underlying connection between nodes, we refer to this connection as **"hidden connection"**. Two

core nodes form a hidden connection if they are not connected and their Semantic Summaries satisfy the threshold based on the search (relaxed / rigorous). A priority queue is used in a DFS implementation to determine whether two nodes are connected directly or indirectly.

Algorithm 3 Find Overlapping Summaries

Input a path hierarchical drawing Γ_s with the Semantic Summaries, a threshold percentage, a boolean value denoting if the algorithm should search for rigorous connections or not
Output Pair of summaries that satisfy the overlapping threshold

```
for every Semantic Summary S do
    Let P be the Semantic Path of S
    for every other Semantic Summary T in P do
        if S.LastNode doesn't collide with T.firstNode then
            move S to the next Summary in P.
        else
            Let S_e ←the summary that has min(Y(S.LastNode), Y(T.LastNode))
            Find common nodes starting from T.FirstNode and ending at S_e.LastNode
            if Common nodes between S & T satisfy the overlapping threshold then
                Use DFS with Priority Queue to check if the core nodes form a hidden connection.
            end if
        end if
    end for
end for
```

In the figure below, a simple hidden connection example is demonstrated, where nodes 16 and 9 form a hidden connection as Target nodes in two semantic summaries containing the nodes <6,4,2>.

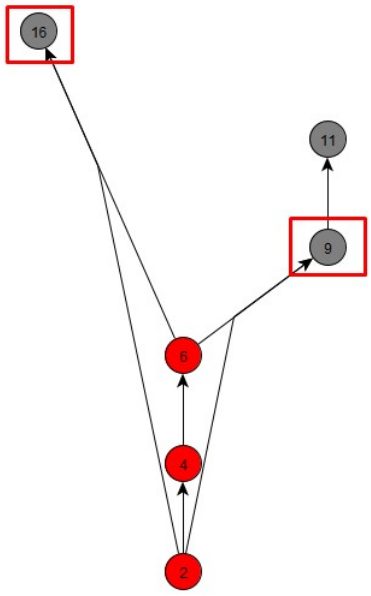


Figure 4: A simple Hidden Connection example

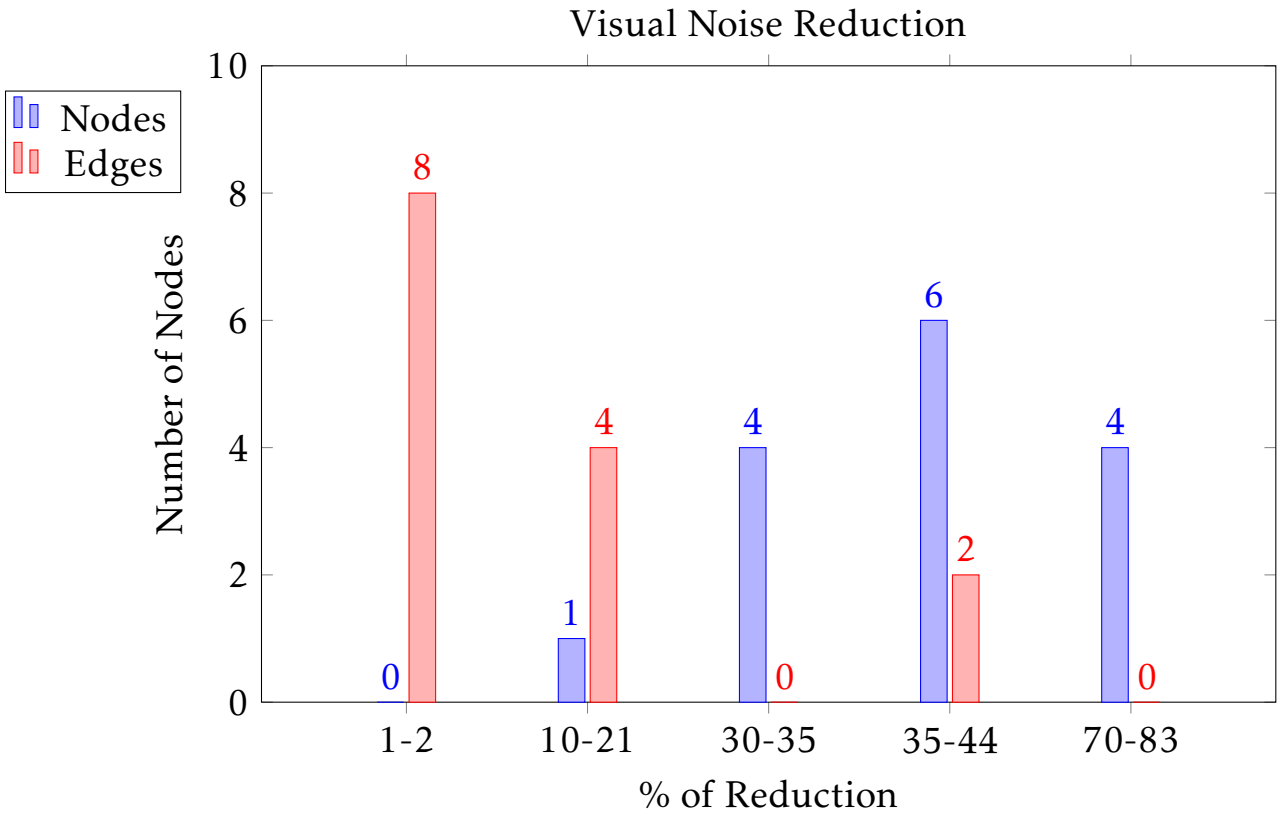
Because the two semantic summaries in the example below are identical, they satisfy any threshold given by the user.

5 Results Comparison

The bar chart below provides information about the reduction of nodes and edges on large graphs after defining Topology Neighborhoods. Nodes appear in blue colour, whereas edges appear in red. The y-axis demonstrates the number of graphs referring to each percentage. While the x-axis demonstrates the percentage of reduction.

Nodes present outstanding results, averaging 48% reduction and reaching a peak at 83%.

On the other hand, edges present less satisfying results, since the majority of the graphs have less than 2% reduction. However, the data sets tested on this technique contained few dense graphs. More precisely, in high-density graphs the reduction of edges reach satisfying results, as demonstrated in the bar chart (35-44%).



6 Feedback

Here, we demonstrate how the number of paths and the percentage of completeness influence the nodes and edges reduction. Percentage of nodes reduction (PNR) is calculated like so:

$$PNR = \frac{N_a - N_b}{N_b}$$

where N_a is the number of nodes after Topology Neighborhoods and N_b is the number of nodes in the PBF graph. The same formula is used to calculate edges reduction.

Number of Paths

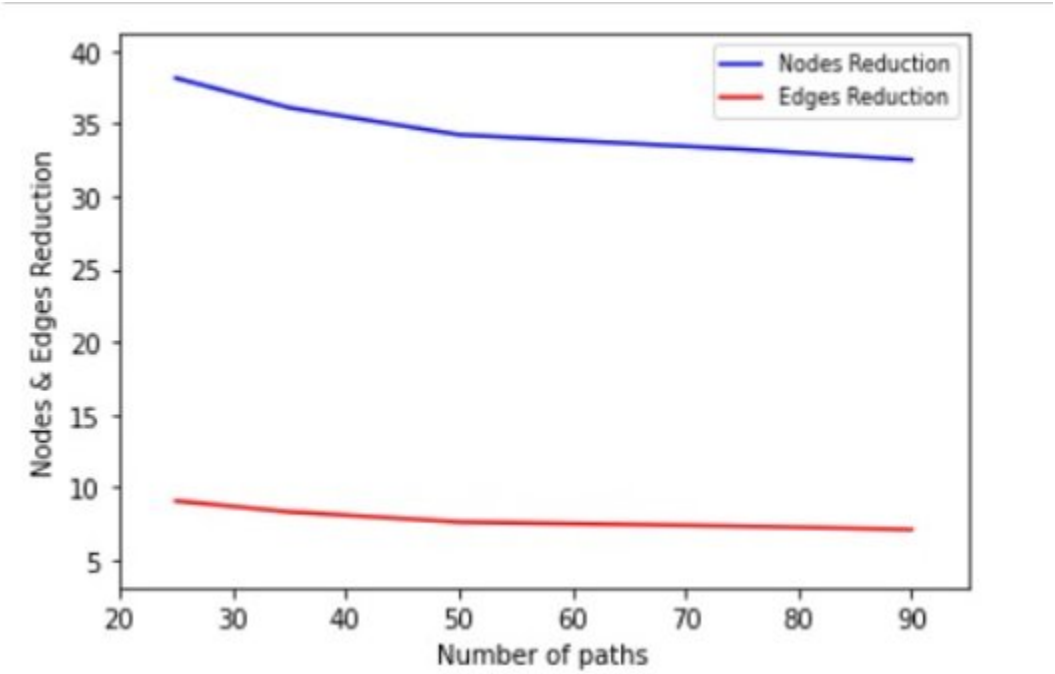


Figure 5

As it can be seen in the figure above, the number of paths have a low or non-existing influence in the nodes and edges reduction.

Percentage of Completeness

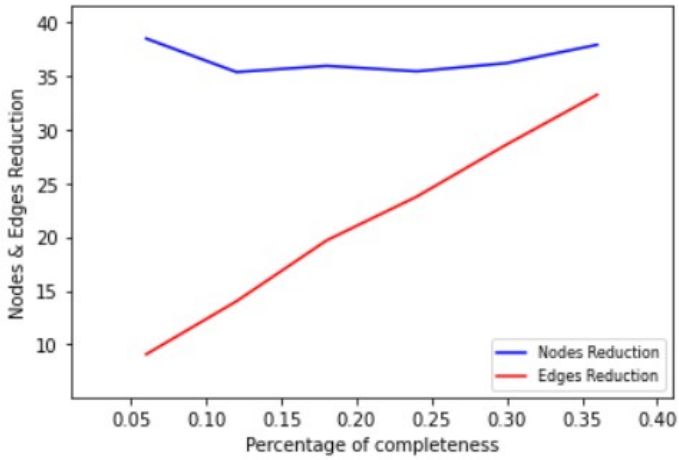
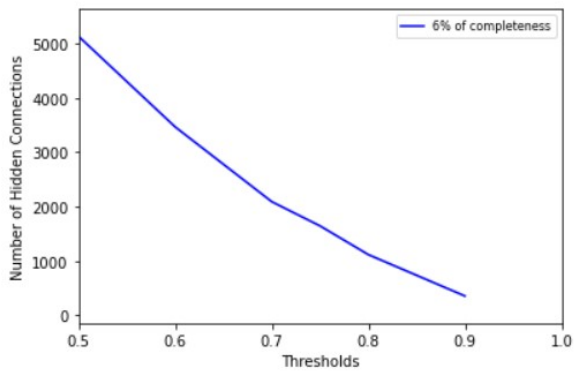


Figure 6

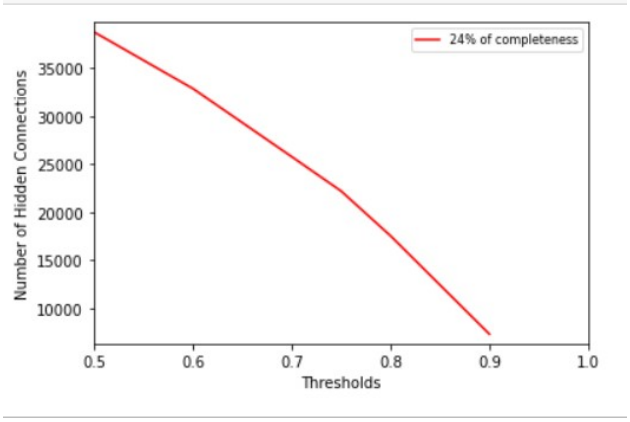
Figure 6 demonstrates how the percentage of completeness influences nodes & edges reduction. The nodes reduction percentage appears in blue, whereas the edges reduction percentage appears in red. In more depth, it seems like the percentage of completeness does not influence the node reduction. However, there is a steady increase in edges reduction as the percentage of completeness increases. Thus, we can say that the percentage of completeness highly influences the edges reduction number.

Threshold

Furthermore, statistics about how overlapping summaries' threshold influences the number of hidden connections are presented in the two figures below.



(a)



(b)

Figure 7

Figure 7(a) shows the influence threshold as it applies to a graph that is 6 percent complete, whereas Figure 7(b) shows the influence threshold as it applies to a graph that is 24 percent complete, which is denser. As the threshold rises, the number of hidden connections gradually decreases in both figures.

7 Statistics

The statistics for some sample graphs are shown in the table below. Those statistics include threshold, percentage of completeness (POC), number of hidden connections, number of semantic summaries created in the graph and the number of semantic pairs compared in order to find hidden

connections.

Table below demonstrates numbers related to semantic summaries and hidden connections.

Threshold	Nodes	P.o.C	hidden_connections	semantic_summaries	semantic_pairs	sem_time (ms)
0.6	7001	0.01	86511	46517	7778216	28876
0.6	7001	0.005	6777	10523	377602	2351
0.6	7001	0.0025	519	2440	19457	376
0.6	10001	0.01	307843	112006	36733762	246451
0.6	10001	0.005	22387	25496	1653174	8454
0.6	10001	0.0025	1744	5668	80067	1083
0.6	15001	0.01	1294838	305219	215450169	3062782
0.6	15001	0.005	82930	66125	8356360	73183
0.6	15001	0.0025	6834	15192	415393	3937
0.8	7001	0.01	25564	46517	7778216	10605
0.8	7001	0.005	1937	10523	377602	948
0.8	7001	0.0025	127	2440	19457	249
0.8	10001	0.01	90694	112006	36733762	86208
0.8	10001	0.005	6448	25496	1653174	4383
0.8	10001	0.0025	506	5668	80067	579
0.8	15001	0.01	381627	305219	215450169	1083303
0.8	15001	0.005	23775	66125	8356360	25442
0.8	15001	0.0025	1984	15192	415393	1914

Whereas, this table shows nodes and edges reductions.

threshold	nodes_before	nodes_after	nodes_perc	edges_before	edges_after	edges_perc
0.6	7001	4080	41.72261	243710	237917	2.3770056
0.6	7001	3433	50.964146	122179	118457	3.04635
0.6	7001	2642	62.262535	61162	57712	5.6407576
0.6	10001	5975	40.255974	497388	486192	2.250959
0.6	10001	5111	48.89511	249363	243274	2.4418218
0.6	10001	4138	58.624138	124814	119943	3.902607
0.6	15001	9393	37.384174	1119276	1098947	1.8162634
0.6	15001	8212	45.256985	561007	550880	1.8051468
0.6	15001	6700	55.336308	280869	273451	2.641089
0.8	7001	4080	41.72261	243710	237917	2.3770056
0.8	7001	3433	50.964146	122179	118457	3.04635
0.8	7001	2642	62.262535	61162	57712	5.6407576
0.8	10001	5975	40.255974	497388	486192	2.250959
0.8	10001	5111	48.89511	249363	243274	2.4418218
0.8	10001	4138	58.624138	124814	119943	3.902607
0.8	15001	9393	37.384174	1119276	1098947	1.8162634
0.8	15001	8212	45.256985	561007	550880	1.8051468
0.8	15001	6700	55.336308	280869	273451	2.641089

8 Pros & Cons

Topology Neighborhoods

In this technique there is a significant reduce in the visual complexity of the graph, since we group nodes and we reduce the edges. However, by doing this, information is lost from the graph. For example, if a super node is created, and an edge reaches that super node, there is no possible way to know to whom this edge refers to inside the super node (vice versa).

Semantic Summaries

Using Semantic Summaries, nodes are now grouped in a new way. By utilizing this method, we can discover a connection between nodes that are neither directly nor indirectly connected. Contrarily, this method must repeatedly iterate through a huge number of semantic pairs (as illustrated in the graphs), which slows down our approach. This method is helpful in social networks and fraud detection, among other applications.

9 Conclusion

In conclusion, we managed to find a way to reduce the number of nodes and edges without losing important information about the graph. Additionally, we managed to construct a connection between two nodes without them linking in the graph. Last but not least, we exploit our methods as a pre-processing step, which is very useful when we are dealing with graphs that can not be visible at all.