In this section, we detail the algorithms for the proposed extension of the Network algorithm to implement subgraph query processing, as well as other improvements outlined previously in section 4.2. We also describe in detail improved algorithms for processing of induced subgraph query.

The algorithms solve two problems, database creation and query processing, hence we divide them into two groups. The first group is concerned with the the construction of a dynamic acyclic graph DAG to store the model graphs. The model graphs are stored in a network with their decomposed subgraphs and the related link and state information. As a detail, we construct a DAG consisting of a set of 5-Tuples, each of which stores information about a graph, its state, its parents, its children and the Edges connecting the children. Creation of the DAG is performed by recursively decomposing the Model graphs and creating a 5-tuple for each resulting graph to store the information. The 5-tuples can be considered as nodes in the DAG as they are connected by a directed edge from parent to child.

The second group of Algorithms concerns the process of detecting subgraph isomorphisms using DAG constructed in previous part. The DAG is the central data structure for this work and is considered a global variant in all the algorithms.

DAG definition First we make a formal definition to help us write the algorithms for the DAG more concisely.

Let $G = \{g_1, \ldots, g_n\}$ be a set of model graph. definition A DAG $D(G)$ is a finite set of 5-tuple (g, s, P, C, E) constructed by decomposition of a set of model graphs G where g is a graph, s is one of $\{unsolved, dead, alive\}$, P is a set of one or more parent graphs $\{p^{'}, p^{''} \ldots\}$, C is a set of two child graphs $\{c^{'}, c^{''}\}$ of graph g, and E is a set of edges connecting the child graphs $c^{'}$ and $c^{''}$. The DAG has to satisfy the following conditions.

enumerate[(1)]

$c^{'}$, $c^{''}$ of graph g *tuple*