Comp 6902 Spring 2018 Project: Reduction

Kyle Nickerson
Arshad Arafat
Tian Wang

June 19, 2018

1 Problem Description

Problem: $K\_COMMON\_SUB = \{< G, H, k >|$ G and H have a common induced subgraph which contains at least k nodes $\}$.

Instance: Two undirected graphs G and H and a natural number k.
Accept if: G and H have a common induced subgraph containing at least k nodes.

2 Reduction

We are able to reduce K_COMMON_SUB to CNF-SAT as follows below. Here we mention the notation used. $G = (V_G, E_G)$ and $H = (V_H, E_H)$ refer to the input graphs, and $S = (V_S, E_S)$ is used to refer to the common induced subgraph of of G and H. Also indices m, n are used to refer to nodes in G, q, r refer to nodes in H, and i, j refer to nodes in S.

2.1 Variables

The variables used are $x_{n,i}$ which signify that node n in G is isomorphic to the ith node in the S, and $y_{r,i}$ which signify that node r in H is isomorphic to the ith node in the S.

2.1.1 Recovery of Original

Given a satisfying assignment to the $x_{n,i}$ and $y_{r,i}$ variables, we recover the original solution as follows.

For $i = 1..k$ check all the $x_{n,i}$ and $y_{r,i}$ terms. There will be 1 of the $x_{n,i}$ and one $y_{r,i}$ term true for each value of i, which indicates that node n in G and node r in H are isomorphic to i in S. Doing this over all values of i gives us the set of vertices in G and in H which form a common induced subgraph. The set of edges in the induced subgraph is composed of all the edges between vertices included the subgraph which were present in the original graph (G or H).

2.2 Constraints

2.2.1 Type 1:

These encode the fact that some node in G must be the *ith* node in S, and some node in H must be the *ith* node in S.

$\forall i \in 1..k$ add constraints

$x_{1,i} \vee ... \vee x_{|V_G|,i}$ and $y_{1,i} \vee ... \vee y_{|V_H|,i}$

2.2.2 Type 2:

These encode the fact that no node in G or H is both the *ith* and *jth* node in the the subgraph ($i \neq j$). To do this, we add the following constraints:

$\forall i, j \in 1..k$ $\forall m \in G$, add constraint:

$\neg x_{m,i} \vee \neg x_{m,j}$

and $\forall i, j \in 1..k$ $\forall r \in H$, add constraint:

$\neg y_{h,i} \vee \neg y_{h,j}$

2.2.3 Type 3:

These encode the fact that if a pair of vertices $m, n \in G$ are isomorphic to the pair $q, r \in H$, and there is an edge between one of theses pairs, there must be an edge between the other pair. To add these constraints we do the following.

$\forall m, n \in G, \forall q, r \in H, \forall i, j \in 1..k$ If there is an edge between exactly one of (m, n) or (q, r), add constraint:

$\neg x_{m,i} \vee \neg x_{n,j} \vee \neg y_{q,i} \vee \neg y_{r,j}$

2.3 Summary

By taking the logical AND of all of constraints listed about, we produce a boolean formula which is satisfiable if and only if our input was in K_COMMON_SUB. Further, because each of our constraints in in CNF form, the logical AND of all of these will also be in CNF.

3 Space complexity of reduction

In this section we look at the effect this reduction has on the size of our problem. Specifically we look at how the inputs to our K_COMMON_SUB problem impact the number of variables and clauses in our CNF expression. The inputs to K_COMMON_SUB are two graphs G and H, and a number k. We refer to the number of vertices and edges in the two input graphs by the variables n_G, m_G, n_H, m_H respectively.

3.1 Variables

In total there are $k \cdot (n_G + n_H)$ variables produced.

3.2 Constraints

3.2.1 Type 1:

In total there are k clauses with n_G variables, plus k clauses with n_H variables produced from the type 1 constraints.

3.2.2 Type 2:

In total there are $k^2 \cdot (n_G + n_H)$ clauses with 2 variables each, produced from the type 2 constraints.

3.2.3 Type 3:

In total there are $k^2 \cdot (n_G^2 \cdot m_H + n_H^2 \cdot m_G - 2 \cdot m_G \cdot m_H)$ clauses with 4 variables each, produced from the type 3 constraints.

An interesting note from this, is that the most clauses will be produced when the input graphs contain about half the maximum number edges possible (ie $m = n^2/2$). With the preferential attachment model we are using for our generator, number of edges in the produced graphs are determined by the parameters given to the generator, and generally the graphs produced are sparse. This will allow us to vary the size of the SAT formula produced, while keeping the size of the graphs constant, an observe how this affects the speed of the solvers, as well as the likelihood of the formula being satisfiable.