# COMP4801 Final Year Project 2017-18

# Interim Report

| | |
|---|---|
| **Project Title:** | **Discovering and Querying Meta-Graphs in Large Heterogeneous Information Networks** |
| **Supervisor:** | **Dr. Reynold C.K. Cheng** |
| **Student Name:** | **Fung Yuet** |
| **UID:** | **-** |
| **FYP account:** | **fyp17045** |
| **Date of Submission:** | **21st January, 2018** |

# 1 Abstract

Heterogeneous Information Network (HIN) is a graphical representation of a dataset. This depiction enables researchers to carry out unprecedented knowledge discovery tasks and to discover unnoticeable but promising facts from the data. Meta-paths and meta-graphs are two meta-level data structures. They were proposed to aid the task of similarity searches and others. However, discovering those structures is difficult manually. Although there has been some effort put into the study of meta-paths generation, most of them cannot scale well enough to handle meta-graphs discoveries. This work therefore investigates the problem of how to discover meta-graphs in large HINs efficiently. A greedy algorithm adapted from an existing framework was proposed in this work to gradually select the most relevant meta-graph and to include it in a regression model. At this stage, the design of the greedy algorithm of generating all potential types has been completed. The implementation of it on discovering a subset of all meta-graphs is under construction. Analysis on the time complexity, space complexity of the algorithm and experiments on how certain variables influence the results will be conducted in March this year.

# Table of Contents


Cover Page i

Abstract ii

Table of Contents iii

List of Figures iv

Abbreviations v

**1** Introduction

**1.1** Background 1

**1.2** Previous works 3

**1.3** Problem statement and objectives 4

1.3.1 Problem Definitions 4

1.3.2 Objectives 5

**2** Methodology

**2.1** Forward Stagewise Structure Generation (FSSG) 6

2.1.1 Theoretical background 6

2.1.2 Demonstration 11

**3** Progress and Schedule 14

**4** Preliminary results

**4.1** Assumption 15

**4.2** Execution result at Phase One 16

**5** Difficulties encountered

**5.1** Impediments to implementing FSPG framework 19

**5.2** Insufficient memory capacity 19

**6** Conclusion 20

Reference 21

# List of Figures


| | | |
|---|---|---|
| Figure 1(a) | HIN | 2 |
| Figure 1(b) | Meta paths | 2 |
| Figure 1(c) | Meta-graph | 3 |
| Figure 2 | A node class hierarchy | 6 |
| Figure 3 | Step 1-3 | 11 |
| Figure 4(a) | Step 4 – part I | 11 |
| Figure 4(b) | Step 4 – part II | 12 |
| Figure 5 | Step 4a | 12 |
| Figure 6(a) | Step 7 – part I | 13 |
| Figure 6(b) | Step 7 – part II | 13 |
| Figure 7 | Schedule | 14 |
| Figure 8 | Examples and counterexample of 2B meta-graphs | 15 |
| Figure 9 | DBLP schema | 16 |
| Figure 10 | Four pairs of positive example | 16 |
| Figure 11(a) | Data Reading – Start | 17 |
| Figure 11(b) | Data Reading – End | 17 |
| Figure 12(a) | Link-only meta-paths generation – Part I | 18 |
| Figure 12(b) | Link-only meta-paths generation – Part II | 18 |

# Abbreviations

| | |
|---|---|
| HIN | Heterogeneous Information Network |
| FSPG | Forward Stagewise Path Generation |
| FSSG | Forward Stagewise Structure Generation |
| LCA | Lowest Common Ancestor |
| BSCSE | Biased Structure Constrained Subgraph Expansion |
| MLAR | Revised model of Least-Angle Regression Model |
| AMPG | Automatic meta-path generation |
| SMPG | Set-expansion meta-path generation |

# 1 Introduction

Background information of the topic is presented in Chapter 1.1. In Chapter 1.2, there is a discussion of some prior works on the related research topics while the problem statement and objectives of this project are introduced in Chapter 1.3 at the end of Chapter 1.

## *1.1 Background*

Data has been increasingly available as the use of the Web and social media by people has been surging. This copious amount of rich information enables the study of knowledge discovery in data aimed for better understanding of the human behaviour. For instance, researchers mine patterns from the training data for the use of trend projection; researchers investigate the topic of relevance between objects to enhance the similarity search. In recent decades, mining in information networks has aroused growing attention from researchers because many datasets can be organized into a graph whose complex structural characteristics allows one to dig out promising knowledge. An information network is a graph *G = (V, E)*, where *V* is the set of nodes (i.e. *objects*) and *E* is the set of edges (i.e. *relationships*) [17]. An example is depicted in Figure 1(a). For each node, it has one or more associated node classes; for each edge, it has one associated edge type. Thus, there are two more important functions: $\tau$: $V \rightarrow \mathrm{A}$ ; $\varphi$: $E \rightarrow \mathcal{R}$. The first function $\tau$ is the node class matching function where each object $v \in V$ matches to one or more node classes $\tau(v) \in \mathrm{A}$. Likewise, the second function $\varphi$ is the edge type matching function where each edge $e \in E$ belongs to one edge type $\varphi(e) \in \mathcal{R}$. Accordingly, there are two kinds of information network:

if $|\mathrm{A}| = 1$ and $|\mathcal{R}| = 1$, it is a homogeneous information network

if $|\mathrm{A}| > 1$ or $|\mathcal{R}| > 1$, it is a heterogeneous information network (*aka HIN*) [14]

Homogeneous information network mining has been studied since the last few decades during which researchers have been developing methods for analysing homogeneous information networks on the task of clustering, ranking and link prediction [14]. It although seems feasible to extend some of these techniques to handle the study of HINs, most of them cannot be directly applied to the problem. It is because the schema of an HIN is far more complicated than the one in a homogeneous information network and this enables an HIN to express richer information. In addition, node classes and edge types are different across objects and relations. Consequently, considering them as identical as those in the case of homogeneous information network loses the semantic meaning and possibly the valuable information one

would have mined from the network. Therefore, researchers should develop a new set of methodologies and principles in studying HINs.

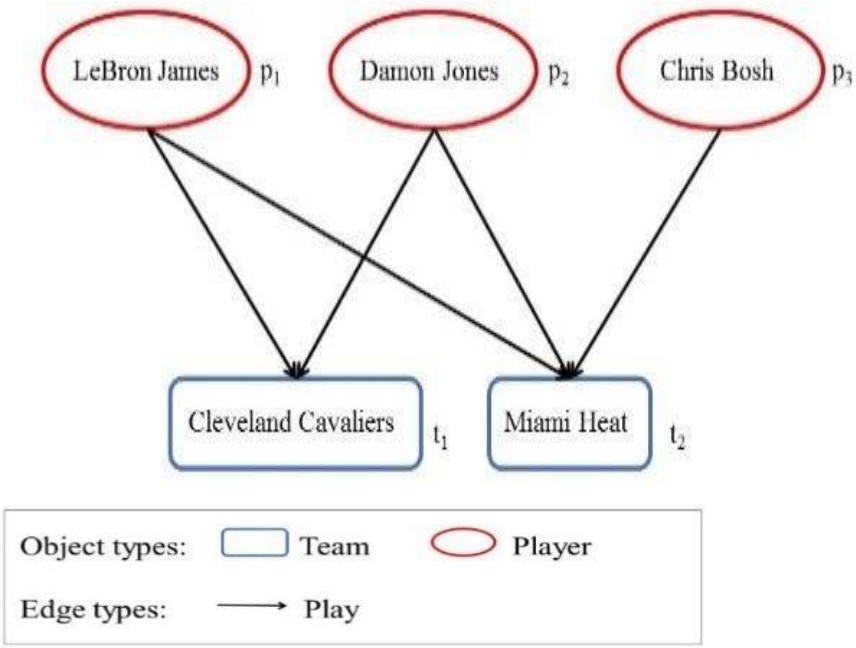


Figure 1(a) HIN

One interesting research topic in HINs analysis is relevance and similarity search. Relevance of two objects defines how similar they are or how tightly they are being related. These research results provoke the analysis of similarity search, classification, clustering and link prediction in HINs [14]. While many works have been done on relevance study, e.g. *Personalized PageRank* [5], *Jaccard's coefficient* [11] and *SimRank* [6], these measures neither consider the different semantic meanings of node classes nor that of edge types. Regarding this issue, Sun. et al [17] proposed the concept of *meta path* and a family of *meta path-based similarity measures.* A meta path is a path comprising of node classes and edge types instead of the actual objects and relations. Node classes and edge types are called meta data in network analysis and thus a path of meta data is recognized as a meta path. Two examples are illustrated in Figure 1(b).

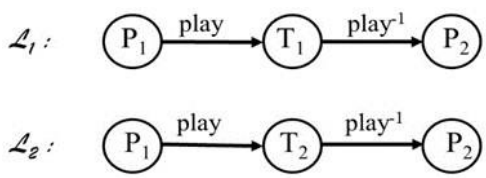


Figure 1(b) Meta paths

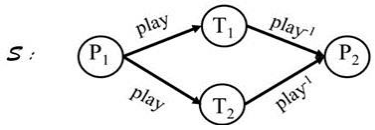


Figure 1(c) Meta-graph

Although meta path has been proved to be useful in many applications, it can only capture simple relationships between the source node and the target one. More importantly, it simplifies all the relations as a single path relation and overlooks the semantic different between a combination of meta paths and a structural relation. Huang et al. [4] subsequently proposed a concept of *meta structure* aiming to provide a data structure depicting complex relations; Fang et al. [2] at the mean time suggested *metagraph*, being a generalisation of meta path and meta structure, to represent various semantic relations. An example of metagraph is shown in Figure 1(c). Meta-graphs (*i.e. meta path, meta structures and metagraph*) have then been proved to be beneficial to product recommendation, community detection, classification and clustering as well as link prediction [1-4,7-8,10].

### *1.2 Previous works*

Despite the substantial studies of meta-graphs application, most of the works assumed these structures are given by experts or are discovered using enumeration or Breadth First Search. Sun et al. [15] assumed the meta paths defining the co-author relationship are given by experts in the study of applying meta paths in prediction and recommendation. Yu et al. [18] and Li et al. [10] both presumed the meta paths will be given in the analysis of recommendation and clustering. Although some researchers attempted to use Breadth First Search to discover the meta path [8,16], they required the user to input the maximum meta path length restricting the search space. Similarly, researchers expected *meta-graph* are given by domain experts [19] or by enumeration [7] for the study of its applications. Since meta-graphs are difficult to define for a complex schema and it is a labour-demanding task [12], while Breadth First Search and enumeration are not efficient for large HINs, it is obvious that researchers should develop an efficient approach to discover these meta-graphs automatically in large heterogeneous information networks.

Some researchers have studied this problem in the last few years attempting to automatically discover and to locate these illustrative data structures in large HINs. Regarding the discovery of meta paths, there are in general two approaches: *example-based training* and *adapted sequential pattern mining*. Example-based training means the user must first provide

positive example pairs. The algorithm framework trains a model based on these pairs and transverses the HIN to discover the meta paths highly explaining the relationship of the given pairs. There are many proposed algorithms using this framework, e.g. FSPG [12], AMPG [1] and SMPG [20], though they differ subtly in some areas. For example, the definitions of the priority score for heuristics pruning and the method of discarding unimportant meta paths. Additionally, Shi et al. [13] proposed a method adapted from well-known knowledge discovery techniques − sequential pattern mining, aiming to simulate mining interesting meta paths as sequential pattern mining. "Generate-and-Discard" suggested by Shi [13] targeted to generate meta paths linking the two target nodes by considering the linkage between the siblings of the two target nodes.

Regarding the discovery of meta-graphs, Fang et al. [2] proposed a heuristic approach to mine meta-graphs from HINs. They suggested to use a set of seed candidate meta-graphs, assuming that two structurally similar meta-graphs are functionally similar. By maximizing the structural similarity of any potential meta-graphs to any seed meta-graphs, it means that the chosen one is structurally and functionally similar to any seed meta-graphs. However, it is unclear whether functional similarity and structural similarity are highly correlated. Moreover, using a set of seed meta-graphs would limit the diversity of candidates. It thus needs more justifications. In short, researchers should develop a more unified and efficient framework in handling automatic discovery of meta-graphs for large heterogeneous information networks.

### *1.3 Problem statement and Objectives*

The goal of this work is to develop a systematic and methodical algorithmic framework allowing efficient discovery of meta-graphs in large heterogeneous information networks. In this work, the method proposed in [12] is adapted with modifications for optimization in the discovery. Below are the definitions of the models used in this study and the specific objectives of this work.

#### *1.3.1 Problem Definitions*

DEFINITION 1 (**HETEROGENEOUS INFORMATION NETWORK**).

*A heterogeneous information network is a graph G containing V, which is the set of nodes (i.e. objects), and E, which is the set of edges (i.e. relationships). There are two more important functions:* $\tau: V \rightarrow \mathrm{A}$ *;* $\varphi: E \rightarrow \mathcal{R}$*. The first function* $\tau$ *is the node class matching function where each object* $v \in V$ *matches to one or more node classes* $\tau(v) \in \mathrm{A}$*. Likewise, the second function* $\varphi\varphi$ *is the edge type matching function where each edge* $e \in E$ *belongs to one edge type*

$\varphi(e) \in \mathcal{R}$. *Note that* $|\mathrm{A}| > 1$ or $|\mathcal{R}| > 1$.

DEFINITION 2 (**META-GRAPHS**).

*Given an HIN G = (V, E, ℒ, ℛ) with* $\tau$ and $\varphi$, *a meta-graph is a graph* $\bar{G} = (\bar{V}, \bar{E})$ *where* $\bar{V} \in \mathrm{A}$ *and* $\bar{E} \in \mathcal{R}$.

PROBLEM 1 (**RELEVANT META-GRAPHS**).

*Given an HIN G = (V, E,* A, ℛ*) with* $\tau$ and $\varphi$, *together with a set of n example pairs* $S_{ep} = \{(s_i,t_i) | i \in [1, \mathrm{n}]\}$ *and a similarity function* $\sigma\sigma(s, t|SS_{mg})$, *discover a set of m meta-graphs* $S_{mg} = \{g_i | i \in [1, m]\}$ *that capture the characteristics of each pair in* $S_{ep}$.

### *1.3.2 Objectives*

In this project, studies are separated into three phases and their corresponding objectives are listed as follows:

- Phase One: Use the existing algorithm framework in [12] to implement a program such that *link-only meta-paths* can be discovered.
- Phase Two: Extend and modify the work in Phase One such that *link-only meta-graphs* can be generated and implement the Lowest Common Ancestor (LCA) lookup.
- Phase Three: Conduct analysis and experiments to understand the performance of the proposed algorithm.

The remainder of this report is arranged as follows. The theoretical information of FSSG is introduced in Chapter 2.1.1. There is a detailed illustration of how the algorithm works for mining meta-graphs in Chapter 2.1.2. The schedule of this study is presented in Chapter 3. Chapter 4 focuses the discussion on the preliminary results obtained from the first phase of the implementation. Theoretical and technical difficulties encountered are discussed in Chapter 5. Conclusion is presented in Chapter 6 with a brief discussion on the future planning for Phase Two and Phase Three.

# 2 Methodology

The proposed algorithm – Forward Stagewise Structure Generation (FSSG) is introduced in this chapter. There are separate analyses on the theoretical information and the algorithm logic.

## *2.1 Forward Stagewise Structure Generation (FSSG)*

In this chapter, there is an in-depth discussion on the theoretical information of FSSG in Chapter 2.1.1. To better illustrate the idea and the flow of logic of FSSG, an example is shown in Chapter 2.1.2 to walk readers through the execution of FSSG.

### *2.1.1 Theoretical background*

FSSG is an example-based training algorithm whose framework was adapted from FSPG proposed by Meng et al. [12]. FSSG incorporates a two-phase framework in discovering metagraphs in large HINs given a set of positive example pairs. In the first phase of FSSG, it generates meta-graphs that have edge types only leaving the node classes empty. Then FSSG fills in the node classes by using the node class hierarchy. A node class hierarchy shows the relationship between various classes. Figure 3 illustrates an instance of a node class hierarchy. In general, FSSG uses greedy strategies to select a set of meta-graphs with high correlation values which denotes their ability in explaining the relationship of every example pairs under a specific similarity measure.

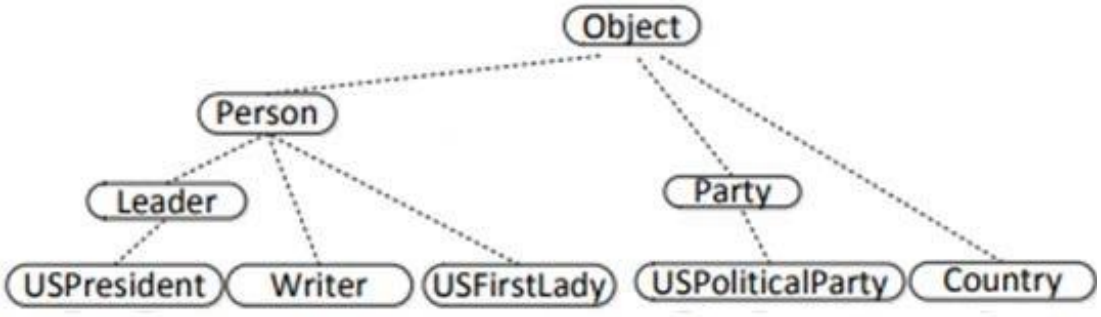


Figure 2. A node class hierarchy [12]

Before executing FSSG, users must provide the program with a set of positive example pairs. These example pairs must be in the form of *(i,j)* where i is the source node and j is the destination node. Given a pair of (*a*,*b*), it means that the user wants to discover meta-graphs that can explain the relationship of *a* and *b*. Given a set of these example pairs, the user would like to locate a collection of meta-graphs that explains the relationship of each pair. For example,

if the user provides two pairs: (*B. Obama*, *M. Obama*) and (*G.W. Bush*, *L. Bush*), the user may want to determine meta-graphs that can explain the relationship of the US President and the US First Lady.

Then, the algorithm generates the same number of *negative* example pairs. It is necessary to include negative example pairs in the discovery process because negative pairs can lower the priority score (i.e. uniqueness) of popular meta-graphs. And thus, the results favour unique meta-graphs instead of common meta-graphs. Next, the algorithm initializes a data structure called **GreedyTree** with a root node containing all the source nodes, their similarity score and the priority score. Here is a short discussion of the two scores. First, a similarity score function determines how close two entities *s* and *t* are related given a set of meta-graphs. Applying this function allows the algorithm to select the best set of meta-graphs in explaining all the example pairs. Below is the equation of similarity function $\sigma\sigma$.

$$\sigma(g, i\,|S, t) = \frac{1}{|\rho(g, i|S, G)|^{\alpha}} \sum_{x \in \rho(g,i|S,G)} \sigma(x, i+1|\, S, t)$$

$$\sigma(g, n\,|S, t) = 1 \; if \; \chi(n_d) = t$$

where *S* is a meta-graph, *t* is the sink object, $\rho(g, i|S, G)$ is the set of instances at *(i+1)*-th layer. The similarity score $n_s$ and $n_t$ given a meta-graph *S* is $\sigma(n_s, 1 \mid S, n_t)$. This is the definition of BSCSE function proposed by Huang in his work on meta-structure [4]. Second, the priority score is required to perform a heuristic discovery on the **GreedyTree** and it in fact is an upper bound of the actual correlation value. The true correlation function is defined as the standard cosine function as below:

$$\cos(\boldsymbol{m}, \boldsymbol{r}) = \frac{\boldsymbol{m} \cdot \boldsymbol{r}}{\|\boldsymbol{m}\| \times \|\boldsymbol{r}\|}$$

where **m** is the similarity score vector in which each entry is the BSCSE score of an example pair; **r** is the residual vector which represents the difference between the regression model and the ground truth value [12]. The default difference of positive example pairs is 1 while -1 is set for negative example pairs. Below shows the definition of the priority score.

$$S_c = \frac{\sum_{u+} \sigma(g, i\,|S, v) \cdot \vec{r}(u,*)}{\|\vec{m}\| \times \|\vec{r}\|} \cdot \beta^L$$

where *u* and *v* are the current starting and ending node; $u+$ are all positive example pairs; $\vec{r}(u,*)$ is the maximum value in the residual vector for example pairs starting from u; $\beta$ is a decay factor and L is the current meta-graph layer.

After calculating the similarity scores and the priority score of the root node, FSSG extends the tree by generating all relevant paths/graphs and create a new node to represent each scenario. FSSG continues expanding the tree with the node having the highest priority score. Moreover, if there are some entries in a node where pairs are identical, it signifies that some individual paths collapse at this point. Therefore, FSSG creates a new node to represent the joint. The process of expansion repeats until a tree node reaches some destination entities. If the actual correlation value of that node is the largest, FSSG returns the meta-graph and its similarity score vector to the main programme where the BSCSE score vector will be added to a modified version of the Least-Angle Regression Model (MLAR). This expansion and addition procedure iterates until the residual vector is negligible. That symbolizes that any additional meta-graphs to be included in the MLAR will not improve the model significantly. At this stage, the discovery is completed, and at last, FSSG returns the set of meta-graphs with edge-types and their weight trained in the regression model.

In phase two, the possible node classes that match the empty spots in each graph are being recorded. Since there are many available choices for a single blank, Meng et al. [12] suggested using the node class hierarchy to resolve the issue. He proposed to take the Lowest Common Ancestor (LCA) on a class hierarchy of all possible node classes for one spot. It is because it can maximize the number of example pairs possibly being explained by each candidate. After the bottom-up transversal on the node class hierarchy and the substitution of the LCA for each missing node class, the results are finalized. FSSG incorporates this method in completing the class information in stage two. Note that ExpandGreedyTree is a function that uses a heuristic data structure **GreedyTree** and it discovers the most relevant meta-graph at one stage in the search space efficiently. Below are the pseudocodes of the algorithm. Most of the variables are defined as they are in FSPG allowing readers to reference and understand better. Interested readers may refer to [12] for additional information and explanation on the FSPG framework.

**Algorithm 1** FSSG$(G, S_{ep})$

**Input:** network $G$, example pairs $S_{ep}$
**Output:** meta-graphs $g_{0,\ldots,k}$, weight vector $\mathbf{w}$

```
r ← {1, ..., −1}; w ← 0; k ← 0;
g_0, m_0 ← ExpandGreedyTree(G, S_ep, r);
k ← 0;
while |r| > ε do
    k ← k + 1;
    g_k, m_k ← ExpandGreedyTree(G, S_ep, r);
    X ← X ∪ m_k;
    corr ← cos(m_k, r);
    compute u, γ;
    r ← r − Xuγ;
    w ← w + uγ;
end
return g_{0,...,k}, w
```

**Algorithm 2** `ExpandGreedyTree`$(G, S_{ep}, \mathbf{r})$

**Input:** KB $G$, example pairs $S_{ep}$, residual vector $\mathbf{r}$
**Output:** meta-graph $g$ having the largest correlation with $\mathbf{r}$, and its similarity score vector $\mathbf{m}$
**Data:** pattern tree $T$

```
update priority scores of leaf nodes in T with r;
while T is expandable do
    M ← node with the largest score S_c in T;
    m ← {0, ..., 0};
    foreach tuple p in M do
        if p.(u, v) ∈ S_ep then
            m(u, v) ← σ(g', i | g, v);
        end
    end
    if m has non-zero entry then
        M.S_c ← cos(m, r);
        if M.S_c ≥ max_{T.leaf} S_c then
            g ← the meta-graph from root to M;
            break;
        end
    end
    else
        foreach tuple t in M do
            foreach pair p in t do
                foreach out-neighbor w of p.v on graph G do
                    e ← link type combination from p.v to w;
                    if M has no child N linked by e then
                        create new child tree node N of M;
                    end
                    g ← the meta-graph from root to N;
                    insert tuple ⟨(p.u, w), σ(g', i | g, w)⟩ satisfying the specific link
                    type in the link type combination to N;
                    update N.S_c;
                end
            end
        end
        foreach leaf node n in T do
            foreach entry e in n do
                if some pairs are identical then
                    if T has no this joint j then
                        create new child tree node N under the ancestors of e
                        accordingly;
                        g ← the meta-graph from root to N;
                        compute the new similarity score;
                        update N.S_c;
                    end
                    else
                        insert a new entry;
                        g ← the meta-graph from root to N;
                        compute the new similarity score;
                        update N.S_c;
                    end
                end
            end
        end
    end
end
return g, m
```

### *2.1.2 Demonstration*

A user scenario is presented below to provide a clear illustration of the execution of FSSG.

**User Scenario – Co-authorship**

**Step 1.** Input positive example pairs by user

**Step 2.** Generate negative example pairs
**Step 3.** Initialize **GreedyTree** with a root node

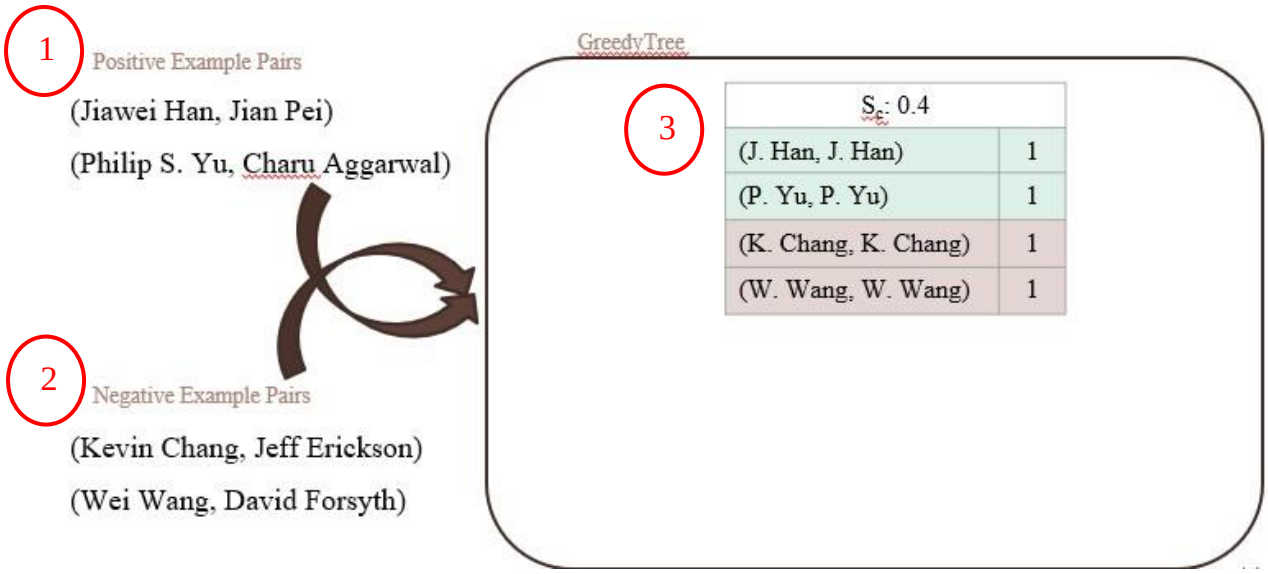


Figure 3. Step 1-3

**Step 4.** Expand the tree with node having the largest priority score

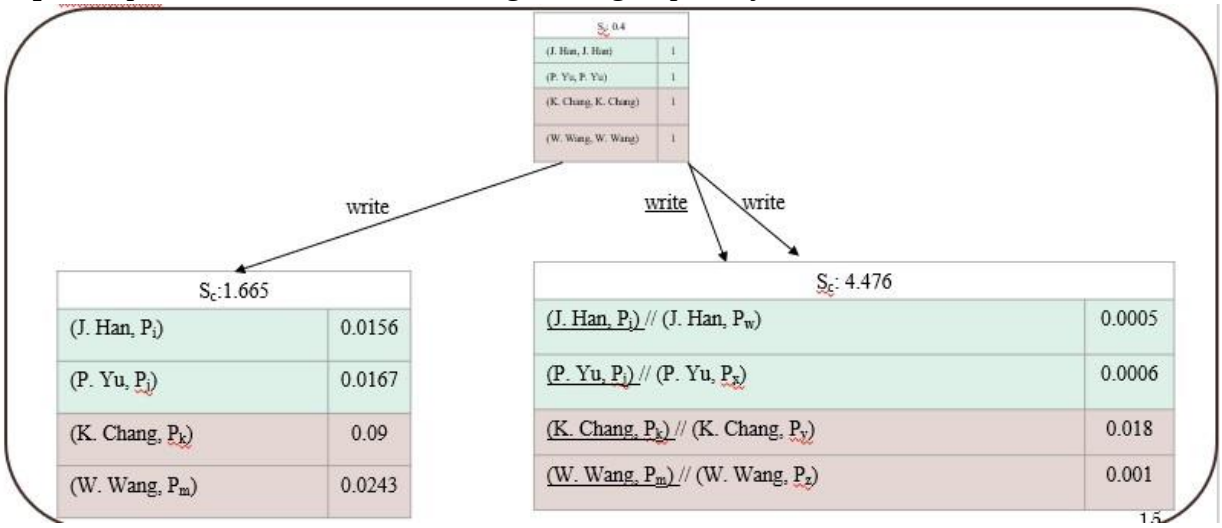


Figure 4(a) Step 4 – part I

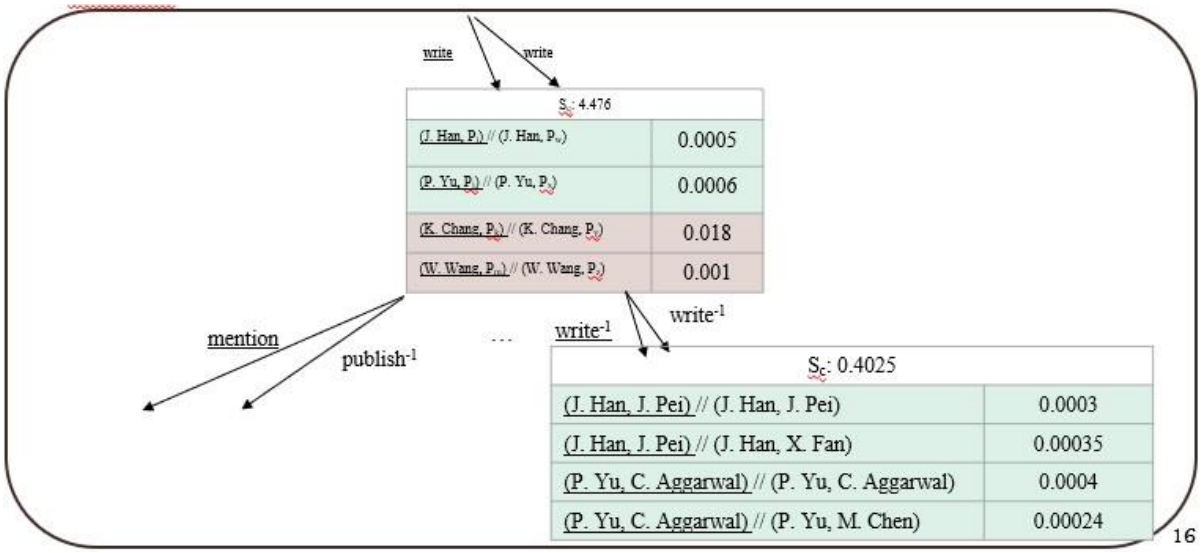


Figure 4(b) Step 4 – part II

**Step 4a.** If some entries containing identical pairs, create a new node

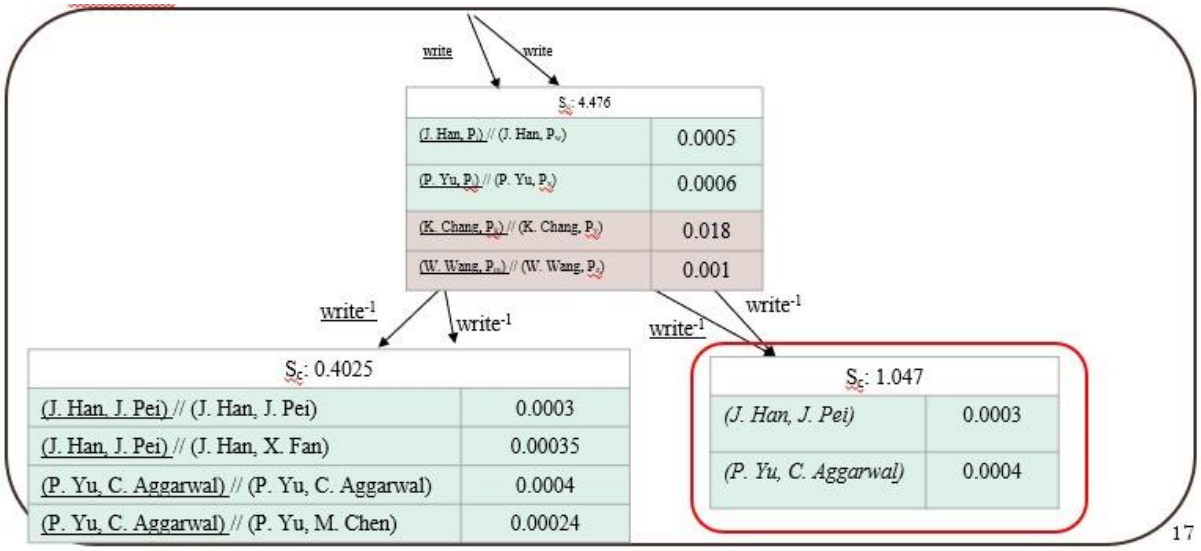


Figure 5. Step 4a

**Step 5.** If a node reaches some sink nodes and its priority score is the largest, returns the meta-graph with its similarity score vector

**Step 6.** Repeat Step 4-5 until the residual vector is negligible

**Step 7.** Perform LCA lookup to fill in class information

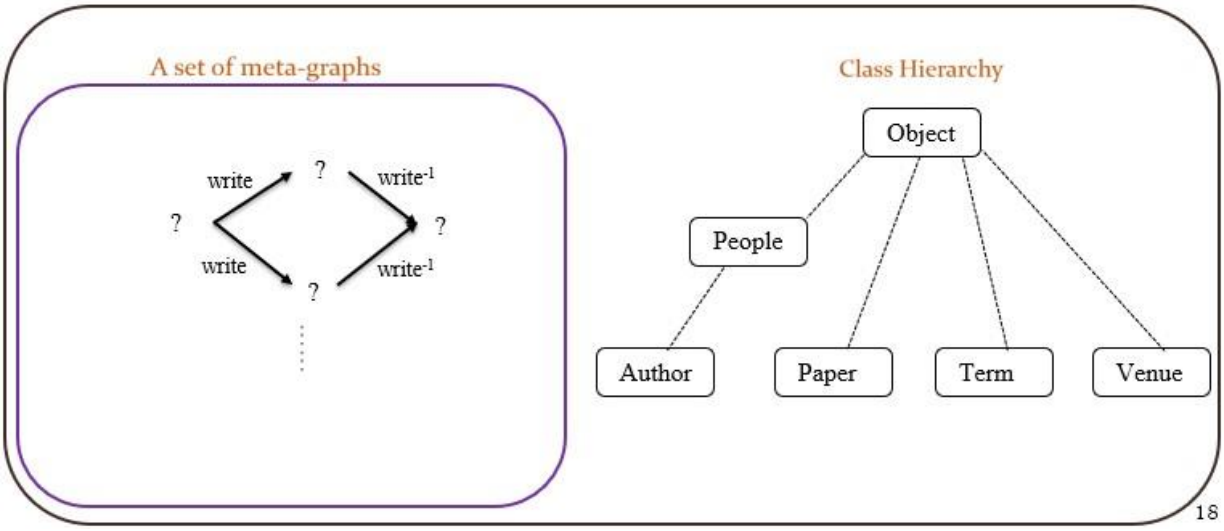


Figure 6(a) Step 7 – part I

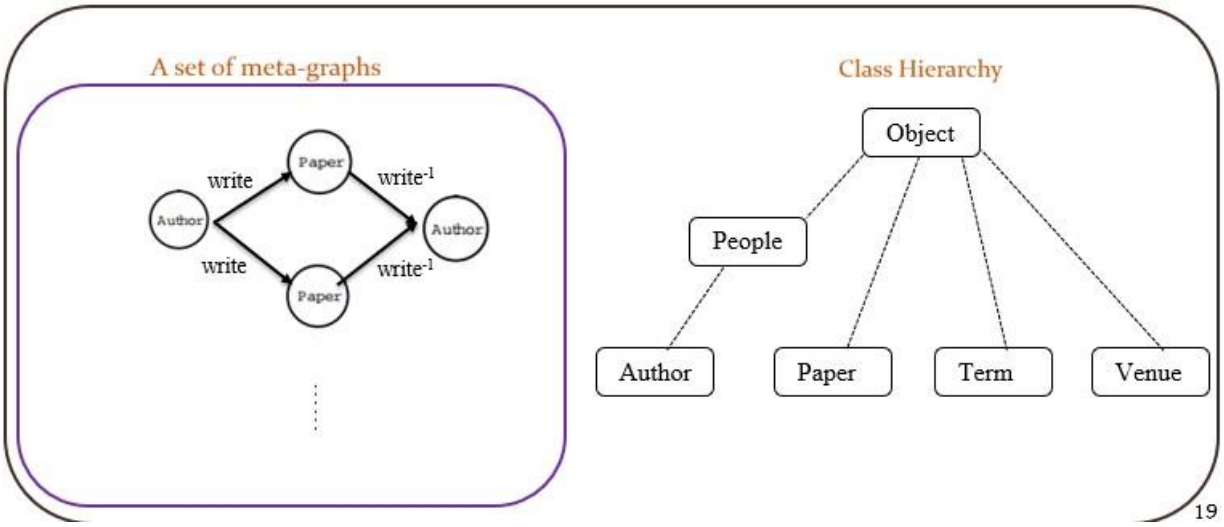


Figure 6(b) Step 7 – part II

# 3 Progress and Schedule

This chapter provides a discussion on the progress of this work and the future schedule.

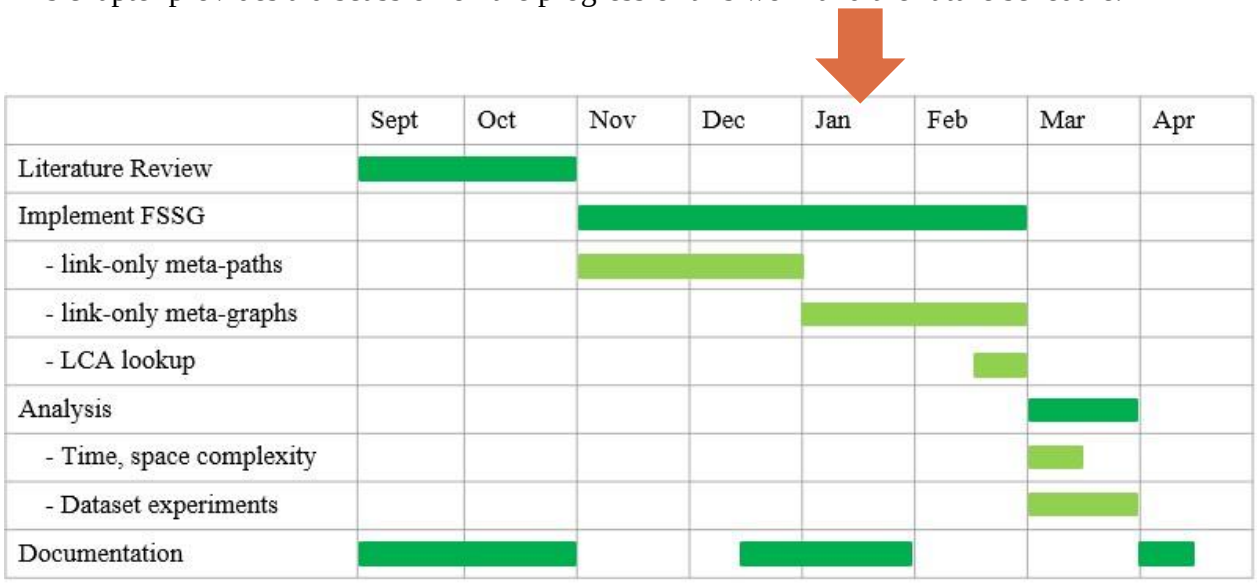


Figure 7. Schedule

Figure 7 illustrates the progress of this project and the time distribution of each subtask. A thorough literature review has been performed during September and October 2017. The major tasks done in this period were to research the background and to design FSSG. This included reading research papers on related topics, comparing and understanding the shortcomings of existing designs, being familiar with the FSPG framework and designing efficient data structures for FSSG. As indicated in Figure 7, the first phase of the implementation focusing on *link-only meta-paths* has been completed by December 2017. It denoted that the programme currently can discover *link-only meta-paths* given a set of positive example pairs.

In the coming months (i.e. January and February 2018), the coding of the phase two of the implementation will be finished, and at the end of February, the programme will be able to discover *meta-graphs* with all necessary information (edge types and node classes). In March 2018, analyses will be conducted to examine the theoretical effectiveness and efficiency of FSSG. More importantly, empirical studies will be carried out to understand the performance of FSSG on real datasets.

# 4 Preliminary results

The first phase of implementation of FSSG was completed. This chapter focuses its discussion on some of the critical assumptions made in Chapter 4.1 and illustrates the output of FSSG executing on real dataset for phase one in Chapter 4.2.

## *4.1 Assumption*

As highlighted by the discussion in Chapter 2.1.1., FSSG is designed to build upon the FSPG framework allowing the algorithm to discover all types of meta-graphs. However, long and complicated meta-graphs are in fact not interesting and not meaningful nor are they easy to generate [12]. Therefore, to facilitate the effectiveness of the algorithm as well as the returned results, FSSG is being implemented in the way that it discovers only *two-branches (2B) metagraphs*. Here is the definition of 2B meta-graphs:

> Given an HIN $G = (V,E)$ and the schema $T_G = (A,R)$, a 2-braches meta-graph is a metagraph $H' = (N,M,n_s,n_t)$, where for all $x \in N$, $out\text{-}degree(x) \leq 2$.

Below diagram shows two positive examples of 2B meta-graphs and one counterexample:

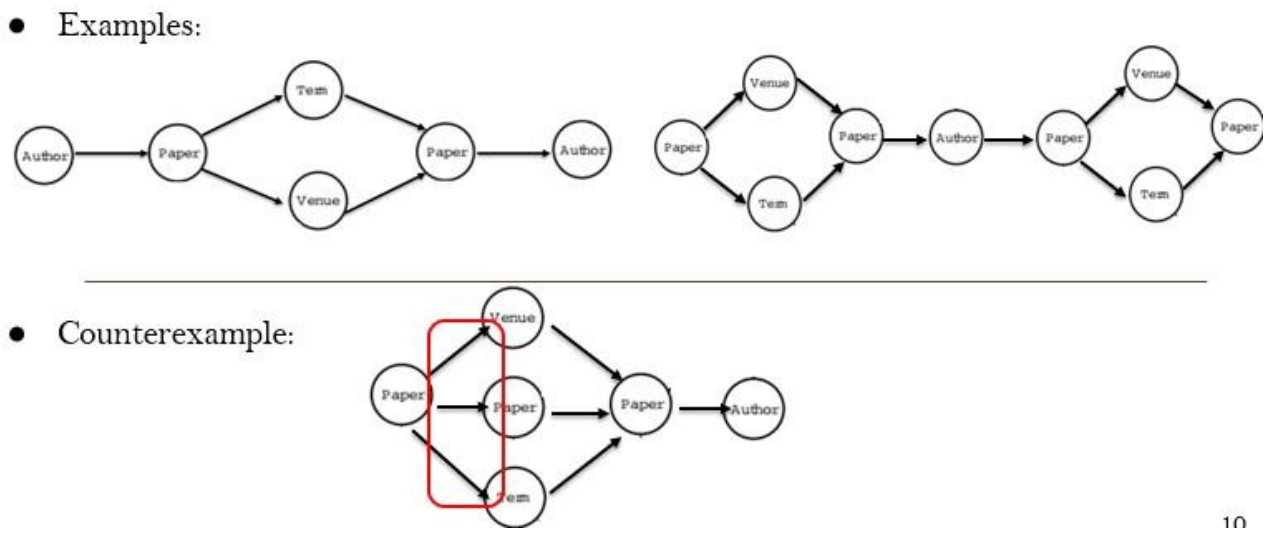


Figure 8. Examples and counterexample of 2B meta-graphs

In short, a node of a 2B meta-graph can perform either no branch or binary branch at every stage of expansion.

### *4.2 Execution results at Phase One*

To demonstrate the result FSSG have achieved, an execution on real dataset was conducted to illustrate the output. One dataset had been chosen for the execution: DBLP four area. This dataset is one of the representative datasets in the study of data mining, network mining and HIN analysis.

DBLP is a bibliography network containing computer sciences journals and conference papers. Since the entire network is enormous and most of the details are not useful for this study, a subset of this network was extracted containing sufficient data for the experimentation. The subset consists of papers published in four research areas: information retrieval, databases, data mining and artificial intelligence [12]. The schema of this subset is shown in Figure 9.

There are four edge types namely writtenBy, mentions, cites and publishedIn. Paper (P), Author (A), Venue (V) and Topic (T) are the four node classes. The subset contains more than 170000 links.

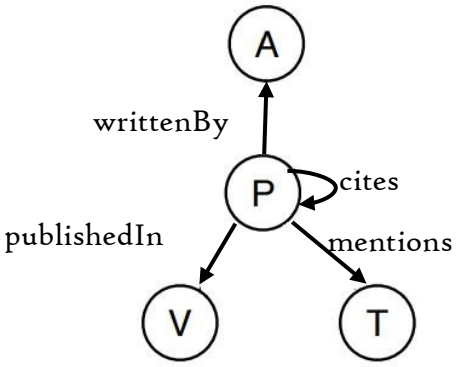


Figure 9. DBLP schema

(Rami Smair, Jamie Callan)

(Wei Wang, Philip S. Yu)

(Jiawei Han, Jian Pei)

(Wei Zhang, Rajat Mukherjee)

Figure 10. Four pairs of positive example

Four pairs of positive example were fed into the program and they are shown in Figure 10. Each pair is an instance of co-authorship, and thus the expecting result should be a set of meta-graphs directly or indirectly explaining co-authorship. Since only the first phase of FSSG has been implemented, the expecting output at this moment instead is a set of link-only metapaths symbolizing co-authorship.

Regarding the setting of variables, as Meng suggested in [12], $\varepsilon\varepsilon$ was set to 0.01 in FSSG. $\alpha\alpha$ was adjusted to 0.5 according to the empirical studies done in [4] and $\beta\beta$ as the decay factor was set to 0.8 [12] to avoid the search going indefinitely. The programme was written in C++ and the execution was conducted on an 4GB memory Win10 machine. The output of the execution is illustrated through Figure 11 to Figure 12.


Figure 11(a) Data Reading – Start

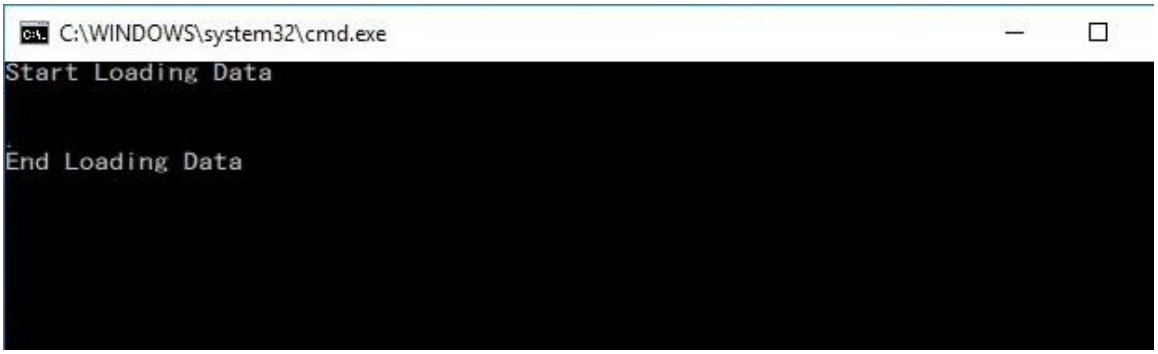


Figure 11(b) Data Reading – End


Figure 12(a) Link-only meta-paths generation – Part I

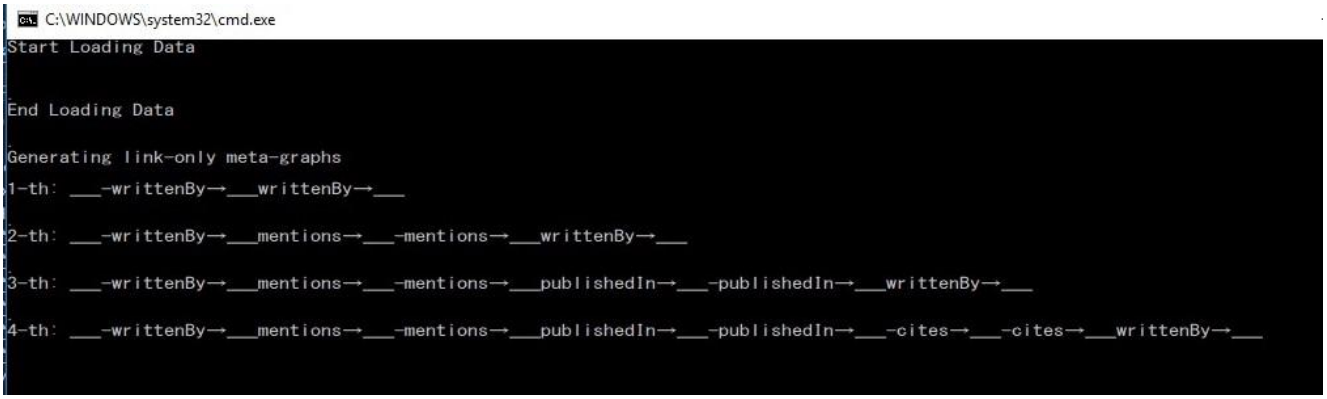


Figure 12(b) Link-only meta-paths generation – Part II

# 5 Difficulties encountered

During the first stage of the project, the team was confronted by numerous theoretical and practical problems. Chapter 5.1 discusses the difficulties in understanding and replicating FSPG framework, and Chapter 5.2 shows an example of technical issue of the execution.

## *5.1 Impediments to implementing FSPG framework*

Since FSSG employs FSPG framework as the main backbone, understanding FSPG thoroughly is required for such extension. However, the description of the FSPG framework in Meng's work [12] is limited and it became difficult to rewrite the FSPG framework from scratch. Although the team fortunately got access to the original coding of the FSPG framework, the authors of the programme did not leave any comments to explain the usage of certain enigmatic variables and the confusing flow of logic. The programme did not have any syntax or compilation errors, it though contains severe runtime errors due to its bewildering and perplexing flow of control. To replicate and to fully extend the framework, the team spent months studying the codes line by line, rewriting the functions, removing unnecessary variables and logic. Despite the delay caused by the unexpected workload, the team has successfully rewritten the FSPG framework as part of the FSSG algorithm by late-December.

## *5.2 Insufficient memory capacity*

Memory capacity inadequacy is a common problem of running experiments in graph mining and network mining field. It is because graphs and networks were embedded with richer information than raw text data. For example, the dataset at the minimum must store all object, all edges, classes of each object, types of each edge and the connections. Lacking memory capacity occurs when the dataset is too large to be loaded into the main memory, the dataset is too large to be searched or the execution requires large amount of memory to support. FSSG indeed requires copious amount of memory to execute because it must store all information of each node on the **GreedyTree** as well as perform cross-checkings. Some potential solutions include running the programme on a server platform providing more memory capacity and designing efficient data structures to facilitate the searching.

# 6 Conclusion

Meta-graph is a useful tool in many knowledge discovery tasks, e.g. similarity search, recommendation. However, currently there has been little research done on studying effective discovery algorithms concerning meta-graphs. The first stage of this work recreates the framework of generating link-only meta-paths given a set of example pairs. FSSG framework was proposed to facilitate the search of relevant meta-graphs. Preliminary execution of the algorithm shows that FSSG is accurate in generating related link-only meta-paths at phase one, and it is likely to return a set of relevant meta-graphs with class information in a reasonable amount of time after the work in phase two.

Even though FSSG did well in the demonstration, there are some limitations of this study. First, most of the parameter values were set by domain experts. Although these suggested values were determined by previous works, it would be beneficial if these values are adaptive to current study or new datasets. Second, although FSSG can effectively mine various types of meta graphs, the running time and the space required when it is deployed in an application would be humongous. More studies are needed to better understand the topic of discovering any types of meta-graphs in large HINs *efficiently*. In the second phase of this study, the work will focus on how to extend the searching to meta-graphs and how to incorporate class information for the results. A comprehensive analysis on the time and space complexity of the algorithm will be conducted in stage three and a rigours experiment on evaluating the influence of some variables will also be performed during that period.

# Reference


[1] X. Cao, Y. Zheng, C. Shi, J. Li and B. Wu, "Meta-path-based link prediction in schema-rich heterogeneous information network", *International Journal of Data Science and Analytics*, vol. 3, no. 4, pp. 285-296, 2017.

[2] Y. Fang, W. Lin, V. Zheng, M. Wu, K. Chang and X. Li, “Semantic proximity search on graphs with metagraph-based learning”, in *ICDE,* 2016.

[3] Z. Huang, B. Cautis, R. Cheng and Y. Zheng, “KB-Enabled Query Recommendation for Long-Tail Queries”, in *CIKM,* 2016.

[4] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis and X. Li, “Meta Structure: Computing Relevance in Large Heterogeneous Information Networks”, in *KDD,* 2016.

[5] G. Jeh and J. Widom, “Scaling Personalized Web Search”, in *WWW,* 2003.

[6] G. Jeh and J. Widom, “SimRank: A Measure of Structural-Context Similarity”, in *KDD,* 2002.

[7] H. Jiang, Y. Song, C. Wang, M. Zhang and Y. Sun, “Semi-supervised Learning over Heterogeneous Information Networks by Ensemble of Meta-graph Guided Random Walks”, *IJCAI,* 2017.

[8] X. Kong, B. Cao, P. Yu, Y. Ding and D. Wild, “Meta Path-Based Collective Classification in Heterogeneous Information Networks”, in *CIKM,* 2012.

[9] N. Lao and W. Cohen, “Relational Retrieval using a combination of path-constrained random walks”, *Machine Learning,* 2010.

[10] X. Li, Y. Wu, M. Ester, B. Kao, X. Wang and Y. Zheng, “Semi-supervised Clustering in Attributed Heterogeneous Information Networks”, in *WWW,* 2017.

[11] D. Nowell and J. Kleinberg, “The link-prediction problem for social networks”, *J. Assoc. Inf. Sci. Technol.,* 58(7), 2007.

[12] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang, “Discovering meta-paths in large heterogeneous information networks,” in *WWW,* 2015, pp. 754–764.

[13] B. Shi and T. Weninger, “Mining interesting meta-paths from complex heterogeneous information networks,” in *ICDM-MODAT,* 2014.

[14] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han, “Co-author relationship prediction in heterogeneous bibliographic networks,” in *ASONAM,* 2011, pp. 121–128.

[15] Y. Sun and J. Han, *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers, 2012.

[16] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla, “When will it happen? Relationship prediction in heterogeneous information networks,” in *WSDM,* 2012, pp. 663–672.

[17] Y. Sun, J. Han, X. Yan, P. Yu, and T. Wu, “Pathsim: Meta path-based top-k similarity search in heterogeneous information networks,” in *VLDB,* 2011, pp. 992–1003.

[18] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han, “Recommendation in heterogeneous information networks with implicit user feedback,” in *RecSys,* 2013, pp. 347–350.

[19] H. Zhao, Q. Yao, J. Li, Y. Song and D. Lee, “Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks”, in *KDD,* 2017, pp. 634-655.

[20] Y. Zheng, C. Shi, X. Cao, X. Li and B. Wu, “Entity Set Expansion with Meta Path in Knowledge Graph”, in *PAKDD,* 201, pp. 317-329.

[21] F.M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledhe. In *WWW,* New York, 2007.