Extracting Fuzzy Gradual Patterns from Property Graphs

Faaiz Hussain Shah, Arnaud Castelltort, Anne Laurent

Extracting Fuzzy Gradual Patterns from Property Graphs

Faaiz Shah
LIRMM
University of Montpellier, CNRS
Montpellier, France
faaiz.shah@lirmm.fr

Arnaud Castelltort
LIRMM
University of Montpellier, CNRS
Montpellier, France
arnaud.castelltort@lirmm.fr

Anne Laurent
LIRMM
University of Montpellier, CNRS
Montpellier, France
anne.laurent@lirmm.fr

***Abstract*—A property graph in a NoSQL graph database engine provides an efficient way to manage the data and knowledge due to its native graph-structure storage. A property graph is a labeled directed graph having nodes and relationships with a set of attributes or properties in form of (key:value) pairs. In this work, we aim at mining such graphs in order to extract frequent gradual patterns in the form of "the more/less A_1,..., the more/less A_n" where A_i are information from the graph, should it be from the nodes or from the relationships. In order to retrieve more valuable patterns, we consider fuzzy gradual patterns in the form of "The more/less the A_1 is F_1,...,the more/less the A_n is F_n" where A_i are attributes retrieved from the graph nodes or relationships and F_i are fuzzy descriptions. For this purpose, we introduce the definitions of such concepts, the corresponding method for extracting the patterns, and the experiments that we have led on synthetic graphs using a graph generator. We show the results in terms of time and memory consumption.**

***Index Terms*—Fuzzy Gradual Patterns, Property Graphs**

I. Introduction

Gradual Pattern (GP) extraction is the process of discovering knowledge from databases as comparable attributes of co-variations. In linguistic expression it may be represented as, the more the value of X_i,. . . , the more the value of X_n, where $i = 1, 2, 3...n$, and $X_1, X_2, \ldots X_n$ are ordinal attributes [1]. The applications of gradual patterns mining can be found in various fields ranging from analyzing client databases for marketing purposes, analyzing patient databases in medical studies, analysis of climate and environment change, etc.

A Property Graph (PG) refers to a data model in which data has (key:value) pairs. A property graph data model enables us to represent the data in natural way in form of graph structure where vertices are called as nodes and edges are called as relationships. The nodes and relationships contain properties/attributes in form of (key:value) pairs. A node can have one or more "labels" that define the role of a node. Relationships are directional and have a "type". They link two nodes. A relationship may contain (key:value) properties as nodes. Graph schema of a sample property graph is shown in Fig. 1, with node labels and relationship types.

Fuzzy gradual patterns are of the form "*the more the X is A, the more the Y is B*" [2]–[4]. These patterns express the information about the attributes and their co-variation. This leads to valuable knowledge for experts for decision making. The task of extracting fuzzy gradual patterns is challenging due to the semi-structured nature of property graphs. Therefore, in this paper, we present a new approach for the extraction of fuzzy gradual patterns from property graphs. We consider fuzzy patterns in order to extract more valuable patterns [5], [6].

The paper is organized as follows: Section 2 describes the preliminary concepts and definitions, including fuzzy gradual patterns and property graphs. Section 3 describes the proposed approach for mining fuzzy gradual patterns from property graphs. Section 4 reports the experimental results while Section 5 concludes and provides perspective future works.

II. Preliminaries

In this section we recall the preliminary work and relevant definitions related to property graphs [7], gradual patterns [1], [8] and fuzzy gradual patterns [2].

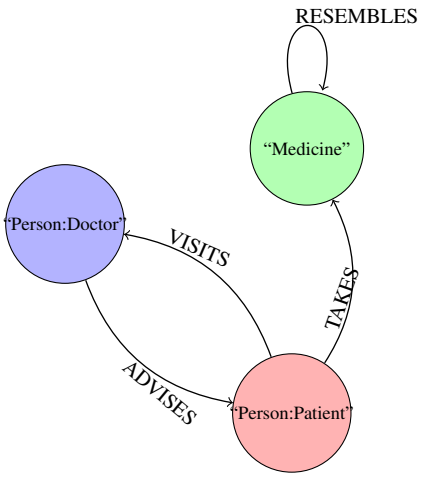

Fig. 1: Graph Schema

A. Property Graphs

A property graph is a combination of nodes and relationships having data in (key:value) format as shown in

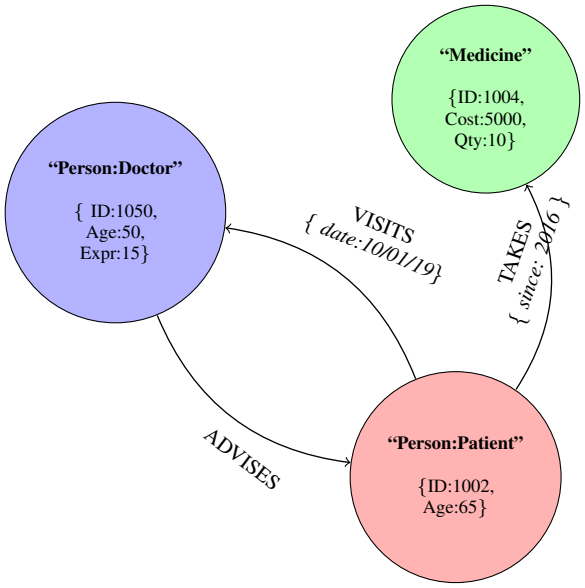

Fig. 2: Property Graph

Fig. 2. We define the property node, property relationship and property graph as follows.

Definition 1 (Property Node): Let Λ_N be a set of node Labels with $NULL \in \Lambda_N$, Π_N be a set of properties where every property $\pi \in \Pi_N$ can take values over a domain $dom(\pi)$. P_N is the set of all possible pairs (π, v) with $v \in dom(\pi)$. A property node n is given by the tuple (id_n, Λ_n, P_n) with

- id_n the unique identifier of n,
- $\Lambda_n \subseteq \Lambda_N$ the set of labels defining the node,
- and $P_n \subseteq P_N$ is the set of properties of n.

Example 1: Λ_N ={Person:Doctor, NULL}, Π_N = {ID, Age, Expr}, n_1 = (2812, { Person, Doctor}, {ID : 1050, Age : 50, Expr : 15}) is a property node as shown in "Fig. 2" for label Person that also carries a label Doctor.

Definition 2 (Property Relationship): Let N be a set of property nodes, Λ_R be a set of relationship types with $NULL \in \Lambda_R$, Π_R be a set of properties where every property $\pi \in \Pi_R$ can take values over a domain $dom(\pi)$. P_R is the set of all possible pairs (π, v) with $v \in dom(\pi)$. A property oriented relation r is given by the tuple id_r, n_1, n_2, λ_r, P_r with

- id_r the unique identifier of r,
- $n_1 \in N$,
- $n_2 \in N$,
- $\lambda_r \in \Lambda_R$ the Type of the relation,
- and $P_r \subseteq P_R$ is the set of properties of r.

Example 2: Λ_R = ({ VISITS, ADVISES, TAKES, RESEMBLES }, Π_R = {date, NULL}, r_1 = (4213, {Patient}, {Doctor}, {VISITS}, {(date:10/01/19)}) is a property relation as shown in "Fig 2".

Definition 3 (Property Graph): Let Λ_N be a set of node Labels with $NULL \in \Lambda_N$, Π_N be a set of properties of node, P_N a set of (key:value) pairs over Π_N, Λ_R be a set of relationship Types with $NULL \in \Lambda_R$, Π_R be a set of properties of relationship, P_R a set of (key:value) pairs over Π_R.

A property graph PG is given by (N, R) where:

- N stands for a set of property nodes defined over Λ_N and P_N,
- R stands for a set of property relationships defined over N, Λ_R and P_R.

Example 3: The details of property graph PG as shown in "Fig 2" are : Λ_N = {(Person:Patient), (Person:Doctor), (Medicine) }, Λ_R = { VISITS, ADVISES, TAKES, RESEMBLES }, Π_N = {(ID, Age, Expr, NULL), (ID, Age, NULL), (ID, Cost, Qty, NULL)}, Π_R = {(date, NULL), (since, NULL)}.

Source Label	Relationship Type	Destination Label	Relationship Count
Patient	VISITS	Doctor	80
Doctor	ADVISES	Patient	95
Patient	TAKES	Medicine	90

TABLE I: Graph Structure Summary

The structure can be retrieved from property graphs in order to deal with the data in a more efficient way. For this purpose, we consider graph structure summaries as described below [9].

Definition 4 (Property Graph Structure Summary): Let SN_L be a source node label with $NULL \in SN_L$, DN_L be a destination node label with $NULL \in SN_L$, R_T be a unique of relationship type between SN_L and DN_L, and C be the count of relationships between SN_L and DN_L.

A property graph structure summary G_{SS} is given by the tuple (SN_L, R_T, DN_L, C).

Example 4: (SN_L = (Person:Patient) , R_T =[VISITS] , DN_L = ((Person:Doctor), C=80).

Example tuples of structure summary are shown in Table I

B. Fuzzy Gradual Pattern

The concept of gradual item and gradual pattern (gradual itemset) in the context of property graph is defined as follows.

Definition 5 (Gradual Item): Let I be a set of items, $i \in I$ be an item and $\delta \in \{\uparrow, \downarrow\}$ be a variation operator. A gradual item $i\delta$ is defined as an item i associated to a variation δ.

Consequently, a gradual pattern is defined as follows.

Definition 6 (Gradual Pattern or Gradual Itemset): A gradual pattern $P = (i_1\delta^1, \ldots, i_k\delta^k)$ is a non empty set of gradual items. A k-itemset is a pattern containing k gradual items.

Example 5: For example, let us consider the pattern *"the more the age, the more the experience"*, it can be denoted by:

$$P_1 = (Age \uparrow , \ Expr. \uparrow)$$

Fuzzy gradual patterns are of the form " *the more/less* A_1 *is* F_1 , *the more/less* A_k *is* F_k [2]. For instance, for a doctor,

the more the age is "almost 40", the more the experience is "almost high". Every attribute is described using a fuzzy partition. For instance, We may consider two sets namely "Low" and "High" for each property of the node as well as the corresponding relationship. Property "Age" would then be described by the "Age_Low" and "Age_High" fuzzy sets. Similarly, for relationship type "TAKES", we may consider "TAKES_Low" and "TAKES_High" respectively.

III. Mining Fuzzy Gradual Patterns from Property Graphs

This work primarily relates to mining fuzzy gradual patterns (see Section II-B) from property graphs. We first formally define them and then propose our approach to retrieve patterns. Furthermore, we discuss about handing unstructured data for patterns retrieval particularly in the context of property graphs.

A. Fuzzy Property Graph Gradual Item

In this section, we formally define the concept of property graph gradual items. As said in section II-A, a property graph is a combination of nodes and relationships. In this section we define what is a gradual item applied to each of this data structure. Property graph node gradual item defined as definition 7 and property graph relationship gradual item defined as definition 8.

Definition 7 (Fuzzy Property Graph Node Gradual Item): Let Λ_N be a set of node labels with $NULL \in \Lambda_N$, Π_N be a set of properties, P_N a set of (key:value) pairs over Π_N, Λ_R be a set of relationship types with $NULL \in \Lambda_R$, Π_R be a set of properties, P_R a set of (key:value) pairs over Π_R. Let $G_D = (N, R)$ be a graph defined over Λ_N, P_N, Λ_R and P_R. Let $\mathcal{P}_i$ be a fuzzy partition of the domain of property $i \in \Pi_N \cup \Pi_R$ within the fuzzy sets $\{f_j\}_{j\in[1,n_i]}$. A graph data gradual item is a fuzzy gradual item i, f_j, δ where $i \in \Pi_N \cup \Pi_R$, $f_j \in \mathcal{P}_i$ and $\delta \in \uparrow, \downarrow$.

Example 6: (Expr, Expr_Low, $\uparrow$) is a fuzzy property graph node gradual item expressing *"the more the experience is low"*.

Definition 8 (Property Graph Relationship Gradual Item): Let Λ_N be a set of node Labels with $NULL \in \Lambda_N$, Π_N be a set of properties, P_N a set of (key:value) pairs over Π_N, Λ_R be a set of relationship Types with $NULL \in \Lambda_R$, Π_R be a set of properties, P_R a set of (key:value) pairs over Π_R. Let $G_S = (N, R)$ be a graph defined over Λ_N, P_N, Λ_R and P_R. A graph relationship gradual item is a gradual item $i^j\delta$ where $j \in \mathbb{N}$ is the depth of the item, $i \in \Lambda_R$ and $\delta \in \{\uparrow, \downarrow\}$.

For the sake of simplicity, when a property graph relationship gradual item is of depth 1, we can omit to mention the depth, as shown in example 7.

Example 7: (RESEMBLES $\uparrow$) is equivalent to (RESEMBLES ^{1} $\uparrow$) and stands for *the number of resembles increases for a medicine*. (RESEMBLES^3 $\uparrow$) stands that *for a particular medicine, number of resembles of resembles of resembles increases*.

B. Retrieving Fuzzy Patterns

Property graphs can be found in a number of complex graph storage or data processing engines. In the context of this paper, we focus on how to extract them from a graph database. The experiments (see section IV) are based on extracting patterns from Neo4j database which is the leader of graph database engines [7]. We thus consider the Neo4j syntax for related examples in this section.

The main steps to extract gradual patterns from a graph database engine are listed below:

1) Retrieve the graph schema from the engine and for each object (node label and relationship type), retrieve the properties;
2) Fuzzify data;
3) Extract gradual patterns and handling missing data.

The rest of this section gets deeper in each step.

1) Retrieve Graph Schema From the Engine: In a graph database, the data are semi-structured, which means that the database does not have an enforced pre-defined schema. Indeed, some properties may appear in some nodes or relationships and not in other ones. For instance, in Fig. 2, the experience (Expr) does not appear for Person 1002. Thus, graph databases are good for dynamic data representation such as networks. This first step is about retrieving the structure of the graph that could thus be associated with schema mining in the literature [10]. To do so, the database engine is asked to retrieve set of triples, each as a combination of label of outgoing nodes, type of relationship and label of incoming nodes as shown in Listing 1.

Listing 1: Retrieving the Database Schema

```
MATCH (n)-[r]->(m)
RETURN labels(n) as srcLabel,
       type(r) as relType,
       labels(m) as dstLabel,
       count(*) as count
```

2) Fuzzifying Data: In this work, we consider the use of fuzzy partitions over the universes of both the properties (e.g., age, salary for a Person) and the relationships (e.g., number of Visits to a Doctor). For this purpose, the data are transformed from quantities to the degree of membership for every subset of the partitions being considered, as for instance low and $high$ for the age.

3) Extracting gradual patterns and handling missing data: We consider existing gradual pattern mining approaches [1], [8] and extend it with the concept of valid database (*"vdb"*) presented in [11] to compute the support. This approach amounts to locally disable some parts of the database during

the computation of the support through the classical pattern mining algorithms so as to only take into account the available information. At every step, the *vdb* is thus retrieved and then used as the basis for the computation of support. In Algorithm 1, the first step is to initialize the binary matrix for each item(property/attribute) and store them in list *L*. For each matrix that represents a property, the algorithm computes binary order matrix [8] and computes the sum of high bits as well as the missing data that are flagged as ? values. This computation is then used to compute the support and only the items having the support greater than the minimum threshold support are updated in list *L*. This updated list is further used by Algorithm 2 where the Hadamard product of binary AND operation of gradual items is performed and the support is calculated. Finally, the program lists the successful frequent gradual patterns.

Algorithm 1: Mining Gradual Property-based Items in the Presence of Missing Values

Input: Properties, minSupport
Output: List of gradual items having support greater than minSupport

Initialize the matrices for each property and store into list L
for all items in list L **do**
  for all rows of matrix M for listItem L_i **do**
    for all columns of row of M **do**
      if L_i.M[row][col] == 1 **then**
        $sum \leftarrow sum + 1$
      else if L_i.M[row][col]==? **then**
        $card \leftarrow card + 1$
      end if
      if card == L_i.M.length **then**
        $flag \leftarrow flag + 1$
      end if
    end for
  end for
  vdb = (L_i.M.length - flag)
  support = sum / (vdb * (vdb-1)/2)
  if support $<$ minSupport **then**
    remove L_i
  end if
end for
Update the list L with successful gradual items of size-1

This approach has been tested over synthetic data, as presented below.

IV. Experiments

Below are the system specifications for running the experiments. Hardware: Intel Core i7-4610M, 3.00 GHz, quad core processor, 16 GB RAM Operating System: Linux generic kernel 4.4.0-134, Ubuntu 16.04 LTS. Software: Java version 1.8.0_181, Java(TM) SE Runtime Environment

Algorithm 2: Mining Gradual Patterns in the Presence of Missing Values from Gradual Items

Input: List L_i of gradual items, minSupport,
Output: Frequent Property-based Gradual Patterns

while items in list L_i ! $= 0$ **do**
  for all listItem L_i **do**
    for all listItem $L_j = L_i + 1$ **do**
      for all rows of matrix M for listItem L_i **do**
        for all columns of row of M **do**
          $resultM[row][col] = L_i.M[row][col] * L_j.M[row][col]$
          if resultM[row][col] == 1 **then**
            $sum \leftarrow sum + 1$
          else if resultM[row][col] == ? **then**
            $card \leftarrow card + 1$
          end if
          if card== resultM.length **then**
            $flag \leftarrow flag + 1$
          end if
        end for
      end for
      vdb = ($L_i$resultM.length - flag)
      support = sum / (vdb * (vdb-1)/2)
      if support $<$ minSupport **then**
        remove L_i
      end if
    end for
  end for
  Update the list L
end while
Output the frequent patterns

(build 1.8.0_181-b13) or higher Neo4j graph database version 3.4.7 or higher. For all experiments, we used static memory allocation to run Java jar file along with the efficient G1 garbage collection and zero "0" biased locking delay. The jvm flags are: -XX:+UseG1GC -Xmx10G -XX:BiasedLockingStartupDelay=0. Complete source code and program execution protocol to generate the results is available at [12].

Neo4j is a NoSQL Graph database which helps to model nodes and relationships in form of connected data. Neo4j uses a declarative query language (cypher) to retrieve data from the graph database. We used Noe4j as a graph database for our experiments. In order to generate graph data, we used our synthetic graph generator. The graph generator interface is available at [13].

The graphs were generated for three datasets of 1000, 10,000 and 20,000 nodes and same number of relationships for the respective dataset. The details of the datasets is shown

S.No.	Nodes	Labels	Properties	Relationship Types	Missing%
1	1K	3	9	3	17
2	10K	3	9	3	17
3	20K	3	9	3	17

TABLE II: Synthetic Dataset: Nodes Details

A1	..	B1	..	..	C3	R1	R2	R3
25	..	43	..	..	74	1	0	2
45	..	88	..	..	NULL	2	2	1
69	..	NULL	..	..	56	1	0	1

TABLE III: Graph structure: (Node Properties with Relationship Count)

in Table II. The range of values of the node attributes are randomly generated between 1 to 100. Also, due to the schema less nature of property graphs, we introduce missing data by randomly selecting the attributes of nodes. The resulting missingness of data is 17% for all the 3 datasets as shown in Table II. For a particular label, the graph structure of node attributes with relationships count is shown in Table III. This graph structure contains 12 attributes i.e., 9 attributes for each node as (A1, A2, A3, B1,..,C3) and 3 attributes for relationship count as (R1, R2, R3). If there does not exist any relationship for a node, the graph database returns 0 in response to the graph structure cypher query. In there exist a relationship for a node, it returns the count of relationship. For example, the first node in Table III having relation type R2 as 0 means that it does not have any relationship type R2 where there are 2 relationships of type R3 for the same node.

V. Discussion

The plots for two types of experiments i.e., with and without fuzzy subset are shown. The experimental results show that with fuzzy subsets, the time consumption is higher as shown in Fig 3 as compared to without them as shown in Fig. 6. Similarly, peak heap memory consumption is also higher with fuzzy subset Fig. 4 and Fig 7. This is due to the fact that the number of attributes to be considered is multiplied by the size of the partition. We observe an increase in the number of patterns being generated particularly at small support threshold values as shown in Fig. 5 and Fig 8. For example, number of patters with 10K dataset at 0.1 support threshold, we get almost double the number of patterns with fuzzy sets. As we can see in Fig. 5 and Fig 8, in both type of experiments, the number of patterns for all the three datasets i.e., 1K, 10K and 20K are almost same. This is probably due to the fact that the data is generated synthetically using the graph generator. It would be interesting to observe the patterns generation with real graph datasets of different sizes. This is one of the points that may be investigated for optimization of results in term of patterns that are being generated.

VI. Conclusion

In this work, we present the mining of fuzzy gradual patterns from property graphs. The approach has been tested on graph data generated by a synthetic graph generator. As in most of the cases, property graphs are semi-structured and thus contain missing data. Therefore, we propose algorithms for dealing with missing data while extracting the fuzzy gradual patterns. In future work, we plan to extend the algorithm for retrieving the optimal size and form of the fuzzy partition to be considered for every attribute and to test our approach on real world graph databases.

References

[1] L. Di-Jorio, A. Laurent, and M. Teisseire, "Mining frequent gradual itemsets from large databases," in *Advances in Intelligent Data Analysis VIII*, N. M. Adams, C. Robardet, A. Siebes, and J.-F. Boulicaut, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 297–308.

[2] S. Ayouni, S. Ben Yahia, A. Laurent, and P. Poncelet, "Fuzzy gradual patterns: What fuzzy modality for what result?" in *SoCPaR: International Conference of Soft Computing and Pattern Recognition*, Paris, France, 2010, pp. 224–230. [Online]. Available: https://hal-lirmm.ccsd.cnrs.fr/lirmm-00798797

[3] H. Koh and E. Hüllermeier, "Mining gradual dependencies based on fuzzy rank correlation," in *Combining Soft Computing and Statistical Methods in Data Analysis, SMPS 2010, Oviedo, Spain, September 29 - October 1, 2010*, ser. Advances in Intelligent and Soft Computing, C. Borgelt, G. González-Rodríguez, W. Trutschnig, M. A. Lubiano, M. Á. Gil, P. Grzegorzewski, and O. Hryniewicz, Eds., vol. 77. Springer, 2010, pp. 379–386. [Online]. Available: https://doi.org/10.1007/978-3-642-14746-3_47

[4] B. Bouchon-Meunier, A. Laurent, M. Lesot, and M. Rifqi, "Strengthening fuzzy gradual rules through all the more clauses," in *International Conference on Fuzzy Systems*, July 2010, pp. 1–7.

[5] R. Kruse, D. Nauck, and C. Borgelt, "Data mining with fuzzy methods: Status and perspectives," in *In: Proceedings of the EUFIT99*, 1999.

[6] E. Hüllermeier, "Fuzzy rules in data mining: From fuzzy associations to gradual dependencies," in *Combining Experimentation and Theory - A Hommage to Abe Mamdani*, ser. Studies in Fuzziness and Soft Computing, E. Trillas, P. P. Bonissone, L. Magdalena, and J. Kacprzyk, Eds. Springer, 2012, vol. 271, pp. 123–135. [Online]. Available: https://doi.org/10.1007/978-3-642-24666-1_9

[7] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*, 2nd ed. O'Reilly Media, Inc., 2015.

[8] A. Laurent, M.-J. Lesot, and M. Rifqi, "Graank: Exploiting rank correlations for extracting gradual dependencies," in *Proc. of FQAS'09*, 2009.

[9] A. Castelltort and A. Laurent, "Exploiting nosql graph databases and in memory architectures for extracting graph structural data summaries," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 25, pp. 81–110, 2017.

[10] P. Laur, F. Masseglia, and P. Poncelet, "Schema mining: Finding structural regularity among semistructured data," in *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings*, ser. Lecture Notes in Computer Science, D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, Eds., vol. 1910. Springer, 2000, pp. 498–503. [Online]. Available: https://doi.org/10.1007/3-540-45372-5_57

[11] A. Ragel and B. Crémilleux, "Treatment of missing values for association rules," in *Research and Development in Knowledge Discovery and Data Mining*, X. Wu, R. Kotagiri, and K. B. Korb, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 258–270.

[12] S. Faaiz. fuzzy gradual pattern using property graphs. source code. [Online]. Available: https://gite.lirmm.fr/shah/fgpg

[13] ——. Graph generator gui interface. [Online]. Available: https://faaizshah.github.io/graphgenerator/

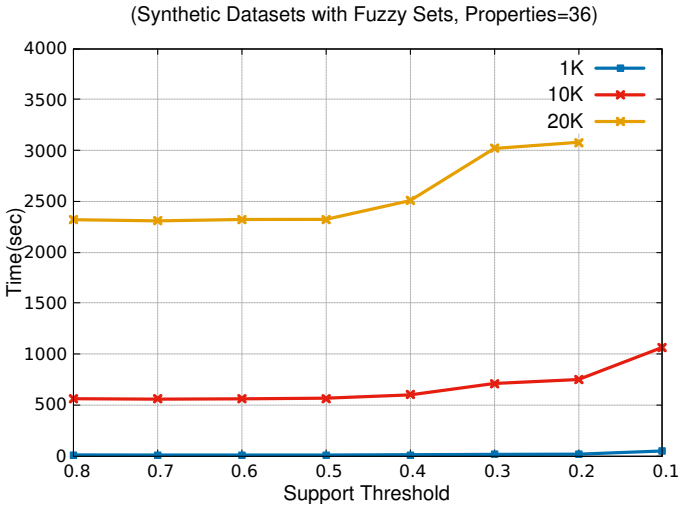

Fig. 3: Time Utilization with Fuzzy Sets

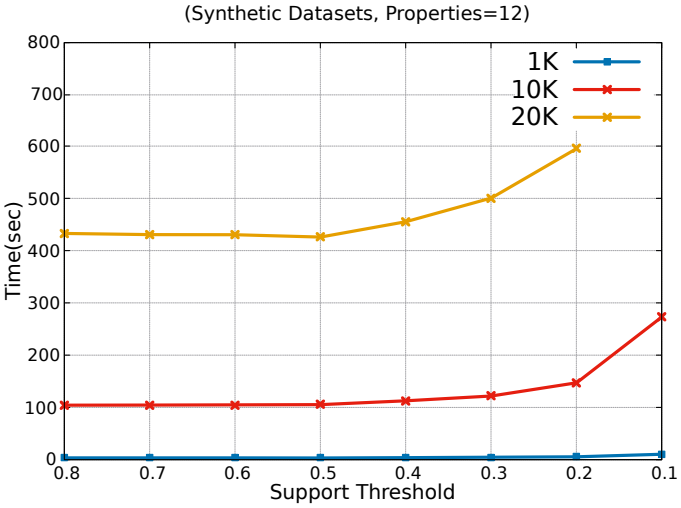

Fig. 6: Time Utilization

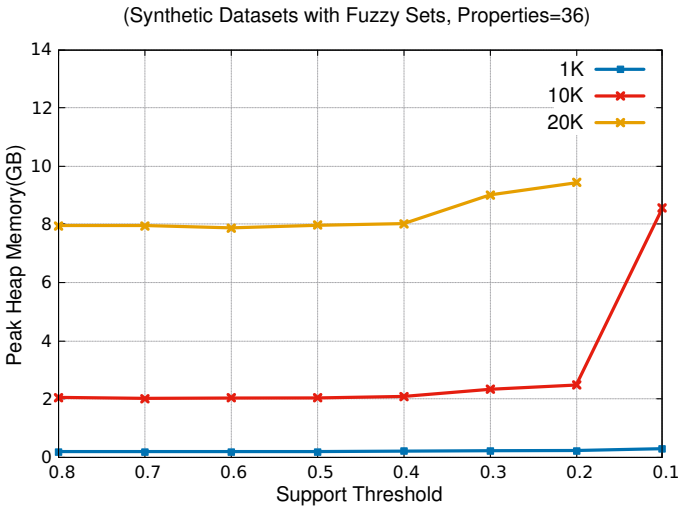

Fig. 4: Memory Utilization with Fuzzy Sets

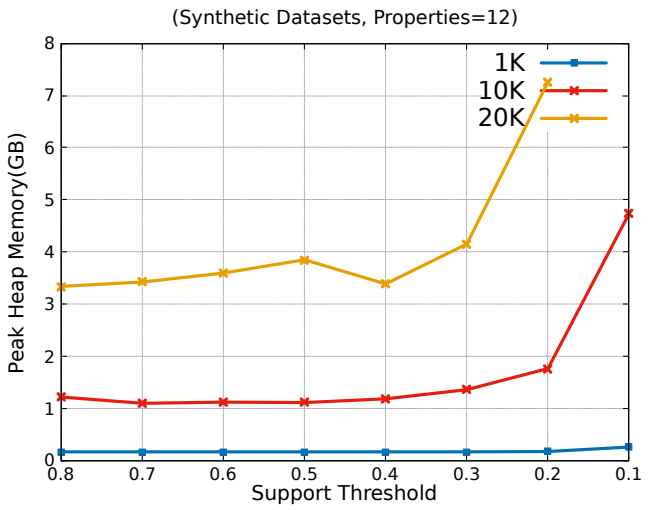

Fig. 7: Memory Utilization

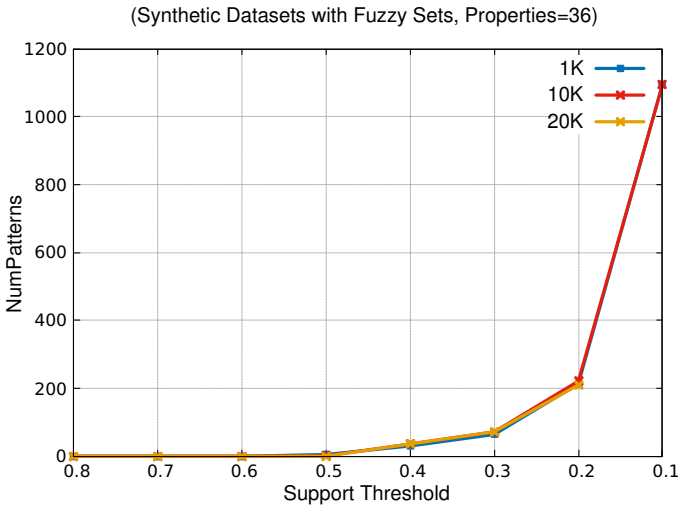

Fig. 5: No. of Patterns with Fuzzy Sets

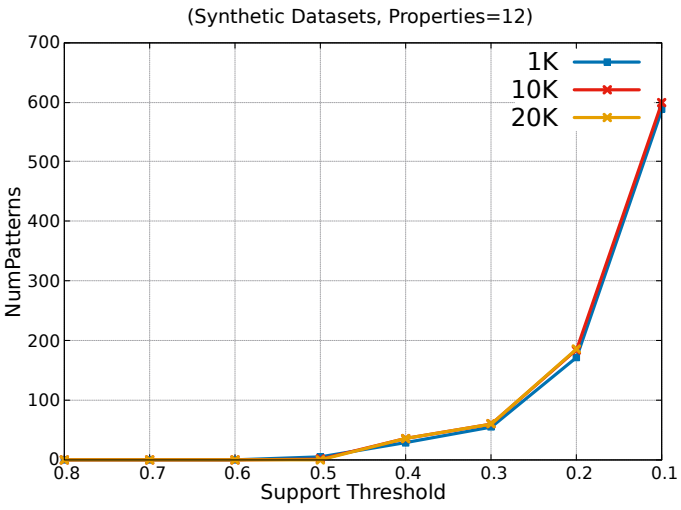

Fig. 8: No. of Patterns