RED-PROFUNDITY DESIGNED FOR TRUST VALUATION IN SOCIETAL NETWORKS

A. Praveen
Associate Professor, Department of Information Technology,
IARE, Hyderabad, Telangana, India

M. Kamala Kumari
Department of Computer Science,
Adikavi Nannaya University,
Rajhamundry, Andhra Pradesh, India

Abstract: *Inter-relationship between two things of similar kind or nature or group for long period of time is defined as Dyadic Communication. Dyad is a basis for Social networks. In general actors in a Social network initiate from a dyadic tie between them. These Social networks range from dyadic to meso and to macro. Analyzing the structure between the entities in a social network reveals the associated patterns existing between them. To describe the structure between two entities there may exist more than one pattern. Considering relations on these path patterns, finding out these structurally different patterns more than one is nothing but redescribing the relations between actors in the network. In this Paper an approach is presented to find those multiple patterns between two related objects. This approach has taken three steps. Initially obtain path patterns for the related pair. Weights are assigned to each path based on the type of relation the actors are bounded to. Weights of these path patterns are estimated and finally paths of similar and nearer weights are selected and found to be as relational redescriptions of the nodes of interest from the given graph. Experiments are done by considering simulated social network with relations and found our approach is efficient than the relational query mining approach.*

Keywords
Relational Redescription, Social Network Analysis, Relations in social networks, Trust Valuation in Social Networks.

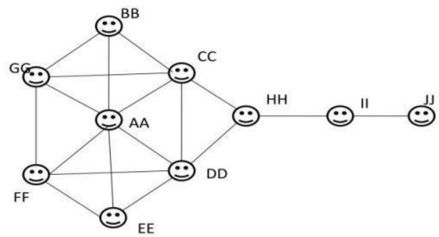
1 Introduction

A. Relational Redescription Mining

Redescription Mining has been playing an important role in domains like Life Sciences, Geology, Cosmology etc., where data is abundant and heterogeneous. Redescription mining identifies the multiple definitions for an object under consideration. Hence redundancy in describing those objects can be reduced. Also establishing relations/links between sets of objects with different structures in between them on the same concept has become vital now-a-days. Different Structures are identified with Redescription Mining, [1]. The concept of this has been extended by considering relations between those objects in a network. Thus relational redescription mining provides a powerful data exploration technique, revealing structurally different connection patterns among objects. For the first time, a social network is considered.

ISSN: 2278 – 1323

Fig. 1 Simple nodes and their connections in Social Network

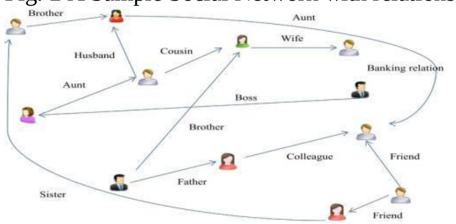


Consider from the Fig. 1, AA is having more number of connections i.e., degree than anyone in the network. If an indirect pair is taken as source and destinations, (GG, II) then they can be described either by (GG, BB, CC, HH, II) or (GG, AA, DD, HH, II). First consider the most promising paths which satisfy the threshold value. The path is constructed or chosen only with the satisfied threshold value which is taken as the relations among paths. The edges which are connected by a relation like parent, sibling, cousins, boss, friend etc., can be considered as weights to that relation. There can be different paths between two nodes in the graph. Then the total cost of these paths which are of minimum difference are taken as redesciption of paths between the given source and destination nodes. Previously, the above said, are represented in the form of Node Link diagrams. These diagrams are of downside when the networks are dense and to discover network structures [2]. A conclusion has been made, if the network is large it takes a long time to identify connections and if it is dense it is impossible. Representation of these networks has been a challenging task for further analysis. Hence only a part of the network is considered. This part is obtained when we confine to a specific domain.

B. Social Network

A basic network arrives when similar objects are connected. A *Relational Social Network* is defined to be a network with the nodes as actors and connected by edges of a relationship such as kinship, friend, financial exchange, dislike or relationship of beliefs, co-authorship, subordinate-ship, ownership etc., Specific group from a social network is obtained when we confine to a specific domain like an organization, a place subdivision, people etc., Network Fig. 2 shows a social network with relations. Any person in the network can be redescribed with the paths emerging or submerging from them. The relations which can be considered as weights describe different ways of links with each other persons.

Fig. 2 A Sample Social Network with relations



Different typical paths from one person to another person describe the relational redescription. Hence relational redescription is described. Given a relational data set, say Actors, DA, their relations DR, their type of relations DT, a redescription mining finds the sets of actors together with their attributes in different terminologies. Redescriptions are extended in terms of Social network paths in which the sets of actors as nodes in the network, are described not only in terms of their attributes, but also in terms of *relations* between them, according to [1]. These relations are characterized by the type and weights are defined accordingly. A path in this relational network can be defined as either strong or weak depending on the total cost of the path.

II. Social Networks Representations with Relational Redescriptions

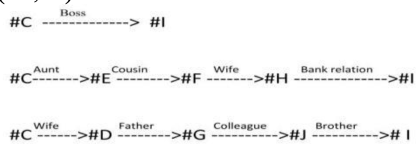
To identify the different paths from a given actor and another, initially find out different paths existing between two nodes in the graph and then obtain the relational redescriptions based on the relationships existing among them. Those relations are weighed according to the type of the relation defined. This task of finding the two structurally different paths that describe the same set of object tuples from a relational dataset is Relational

Redescription mining. From the Fig. 2, which is an example for Social network with relations, we see the actors, *#A, #B….,#L.* The attributes of the nodes can be (male, female). They are linked with *relations* such as Mother, Sister,

ISSN: 2278 – 1323

Cousin, Colleague, Friend etc., To obtain relational redescriptions if object tuples, are given say for example, from the figure (#C, #I) then the paths obtained are shown in Fig. 3. For the first time relational redescriptions are introduced with Social networks as domain. Similarity between relations is measured by defining the type of relations. By defining *Type 1* to be parental relation, *Type 2* to be sibling relation, *Type 3* relative and so on, the total cost of the path would be the sum of these types. This considered as support of that path, P we measure as the *similarity, δ* of two different paths P and P' as:

Fig. 3 Relational redescriptions for the pair (#C, #I)



III. Representation of Relationships in Graphs

Graphs with relationships on edges are represented in the form of Collaboration graphs where this in turn is represented in two other graphs:

Collaboration Graphs

O Balanced Graphs

o Unbalanced Graphs

Collaboration graphs can be used to illustrate good and bad relationships between humans. A positive edge between two nodes denotes a positive relationship (friendship, alliance, dating) and a negative edge between two nodes denotes a negative relationship (hatred, anger). Signed social network graphs can be used to predict the future evolution of the graph. In signed social networks, there is the concept of "balanced" and "unbalanced" cycles.

A. Social Network Analysis

A Social network Analysis examines the structure of Social relationships in a group to uncover the informal connection between people. In a consulting setting these, relationships are often ones of communication, awareness trust and decision making. As an approach to looking at these relationships SNA has been a since longtime, Kate Ehrlich et al.[4].

Most recently, SNA has become an important tool for organizational consultants seeking to understand the connection between patterns of interactions and business outcomes such as job performance, job satisfaction, adoption of new ideas or technologies, likelihood of information getting shared, and creation of new ideas.

B. Relation Types in Social Networks

In this Chapter more than two structurally different path patterns of a given graph which describes the same set of object tuples are considered. Descriptions between two nodes as the path clauses between them which contains sets of in between nodes and relation existing between them are observed. Though the nodes or objects of interest have attributes like male, female those attributes are not considered at this point as they will be explicitly viewed by the relation existing between adjacent nodes. Say for example if the relation existing between #A→#B is sister, it is well known that #A is a female. Consider relations as (Father, mother, brother, sister, aunt, uncle, cousin, boss, colleague, friend, and customer). As the relations are specific more number of relations exists. From the above analysis weights are defined according the relation existing. Final redescriptions are filtered based on the total weights of the paths clauses. The weights that correspond to the relations are as shown n Table 1.

Relational data is viewed as directed graph which contains nodes and edges. Nodes are the objects we map and edges are the relations existing between them. Only edge attributes considered are E = (relation, weight) where relations take the values of the set mentioned above and weights correspond accordingly. Two types of Boolean functions considered are on node and edge attributes. Let V be the set of individuals in the network. Let the node predicate be v(o).This is true when a node exists and a positive object tuple exists in the network. Let the edge predicate be *eW(o1,o2)* and is true for a pair of objects (o1,o2) if and only if the edge label *Ei(o1,o2)* is defined and takes a value from the defined adjacency lists.

Table 1: Relations, types and their assigned weights

Relations	Type	Weight
Father, Mother, Husband, Wife	Type 1	5
Brother, Sister	Type 2	4
Aunt, Uncle, Cousin	Type 3	3
Friend	Type 4	2
Boss, Customer, Colleague	Type 5	1

ISSN: 2278 – 1323

From the social network Fig 4.2, node predicates will be *v(#A), v(#B),v(#C) etc.*, and edge predicate will be *e*Aunt*(#C,#E), e*Father*(#D,#G), e*Friend*(#L, #J) etc*.

Here the graph clause is a definite clause of the form g(V1,(q(Vi,Vj),Vn): vi-b1-b2…-bn-vj where *v1* and *vn* are the starting and ending nodes of the given graph, V1 and Vn are query variables, bi are the edges of the path in between the query variables. The comprised path should be of the regulation that they should be linked such that all edge predicates connects for each pair of query variables say (Va,Vb). A path clause is linked iff for each pair of query variables (Va, Vb) there is a sequence of variables Z0, Zi…., Zk with Z0=Va ,Vb = Zk and for all i= 1…k, there exists, e(Zi,Zi+1), e(Zi+1, Zi+2) ….e(Zk-1,Zk).

According to the above notations, the path clauses between (#C,#I) are as follows:

*G(#A,(#C,#I), #L) : v(#C)-e*Boss*W1(#C,#I)-v(#I) :v(#C)-e*Aunt*W3(#C,#E)-e*Cousin*W3(#E,#F)-e*Wife*W5(#F,#H)-e*Customer*W1(#H, #I)-v(#I) :v(#C)-e*Wife*W5(#C,#D)-e*Father*W5(#D,#G)-e*Colleague*W1(#G,#J)-e*Brother*W2(#J,#I)-v(#I)*

Compute the weight of each path as the value of the weight attribute. Add all these weights of each path clause. These path clauses are nothing but the descriptions of the query variables. Generations of more than one are the redescriptions of the query nodes. Considering relational redescriptions are the ones with the similar path weights which satisfy the given threshold value. Here the threshold value considered is of the path weights only. Number of paths can also be as one of the threshold values. From the above paths, Weight of path 1 is 1,weight of path 2 is 12(3+3+5+1) and weight of path 3 is 13(5+5+1+2). Hence outcome of the query pair (#C,#I) is path2 and path 3 which are relational redescritions of the pair.

C. Relational redescriptions for Social Networks

Relational redescription mining was introduced by Galbrun.E and Angelika Kimmig[1], which finds the two structurally different patterns that describe nearly same set of object tuples in a relational dataset. They experimented in the domain of explaining kinship terms which can be explained with their approach of finding frequent paths and build patterns algorithms. Relational redescriptions when compared to simple redescriptions provide insights in the relational data which identifies patterns that link objects of interest.

The proposed algorithm RED-Depth has been instigated by Social communities of a group. The tasks of finding the paths between two objects, their relativity strength, searching for an object, communication frequency, distance between the nodes, failure of a node, Interdependency between objects, Closeness Centrality, Degree Centrality, Centrality etc. can be analyzed [5],[6]. Analysis of Social networks aims at identifying and emphasizing the social structure and relational aspects of these structures. Hence relational redescriptions utilize these relations rather than attribute data[7],[8]. We have considered for the first time applying relational redecriptions to a Social network. Initially focus is only on identifying different paths between two actors in a given network.

D. RED-Depth Algorithm

In the working example of Section II, relational redescriptions are shown for the tuple (#C , #I). Initially in this section the algorithm is framed by first considering only existence of relation and not focusing on what type of relation it is. First step to address the fundamental problem is as follows: A network or graph is specified which consists of nodes and a set of positive object pairs among them. Existence of an edge between nodes is symmetric i.e., *(a, b)* is same as *(b, a)*. When an arbitrary object tuple is given we need to identify the number of acyclic paths between them.

A two phase approach is designed. In the first phase we construct adjacency lists for each node of the graph.

From source node to the destination node, trace all the paths emanating from source node. A node's adjacent list is explored recursively until target node is found. A formal Depth First Search approach is utilized for each path until it reaches target node. Once the target node is identified for the current paths, a step back is taken and repeats the same process until adjacency list is over. Second phase considers the weights on the graphs. A value is assigned to the relation type and calculates the total strength of each path. Relational redescriptions

ISSN: 2278 – 1323

are finally considered as the path phrases of similar total weight which are definitely of above threshold value. Directional graphs are considered as the relational aspect of one object is not equal to the other object in many cases except few. If *#A* is mother of *#B* and *#B* cannot be mother of *#A.*

E. Path finding in graphs

Path finding is very important for graph analysis. A path is represented as a set of entries, in which the row index of each entry is the same as the column index of the next one; i.e., a path $P = \{ v0, v1, v2\ldots, vn\}$ of graph $G(V, E)$ is equivalent to a series of entries in the corresponding adjacency matrix, $\{e[v0, v1], e[v1, v2], \ldots. e[vn\text{-}1; vn]\}$, where $e[vi, vj]$ is an entry at the intersection of column *vi* and row *v j*. Unlike clusters or star structures, a path can appear as any loose pattern in an adjacency matrix.

F. Constructing Adjacency lists

Let a graph, G = (V, E) consists of a set V of vertices(nodes) and set of edges, E directed or undirected. Any binary relation is a graph, network of roads, circuits representation, communication networks etc. Advantage of using adjacency lists takes only (+) space when compared to Adjacency matrices representation which takes $(\ )^2$ space. Also, a list is easy to find the vertex degree. A balance has to be checked and maintained when traversing and considering a storing a graph between a matrix and a list.

A Graph is represented as an array, ADJ, of lists. The list ADJ[i] contains a list of all the vertices adjacent to vertex i V.

Data structures used will be mostly arrays and stacks as they are the internal representation of lists. I/P: Graph, *G* = (*V, E*) V ← Number of vertices in the graph E ← Set of all positive edges in the graph. [p] ← An array to hold list of all vertices in the current path [x] ← List of arrays

S: Source and D: Destination vertices, W= 0; i=0

O/P: Different unique paths from source to destination, [x]i.

RED-Depth (vertex S)
Add S to the top of the stack
for each vertex,v adjacent (S)
{
If (v== D) then
{
Increment i
[x]i save all elements of stack and display in reverse append D to [x]i; printpath([x]i);
Wj = Wj+Weight(S,D); return
Wj; Wj= Wj-Weight(S, D);
pop top of the stack and reduce that edge weight;
}
else if v is not in the stack
{ Wj=Weight(S, v) + Wj , RED-
Depth(v)
}
return [x]i
}

Weight( ) function finds the edge relation of the given pair and returns the weight of the path based on the relation. These weights are stored in a list, W and are given as input to the WeightSort algorithm to keep them in sorted order.

Finding similar and strong weights of the paths: This can be done by sorting the path weights of all the paths. Using Bubble sort arrange all path weights starting from maximum weight. So that first few paths can be considered and the remaining can be filtered.

Consider the first few path weights filter those paths and leave the remaining paths. The obtained paths of nearer weights are the relational redescriptions of the given nodes of description.

G. Working of the algorithm

The algorithm RED-Depth is a modified one of Depth first search algorithm. In Depth first, search is done for target node starting from a node and a path. Once target node is found, the search is stopped. Even if other paths exists they are not explored. In our approach Depth first is repeated for all the existing paths. A graph G, is taken as input with V number of nodes and E number of edges. Initially an empty stack p is considered to store visited nodes. An array [x] is considered to store paths once the destination node is found. S and D variables store the source and destination nodes given. After the random input source is taken, it is stored in the stack. For each adjacent nodes of the source node are checked whether it is a destination node. Adjacent nodes are obtained for each vertex from the function Adjacency( ). If the node is adjacent then all the elements in the stack are popped and stored in the

ISSN: 2278 – 1323

temporary array and returned with the destination node. At the same tme weight of the path is also stored in Wj and returned. For the alternate paths we pop top of the stack element and continue from there. If the adjacent node is not a destination node, then the weight of the path is updated and the Getpath( ) is called for the adjacent node. Weight( ) is a function which returns the value for weight attribute based on the relation on the edge. Weight values are specified above in Table 1. Different unique paths are returned from source and destination nodes and stored in the arrays of [x]i by incrementing i.

H. Finding relational redescritions

This is outlined in the WeightSort Algorithm. There will be many paths generating from a node in the graph to find all existing ones. This can be filtered by finding all the weights of the paths and remove the ones which are outliers. That is, remove the paths which contains a maximum difference in weights of all the generated ones. If there are more number of paths between the given target nodes, this algorithm finds more than two redescriptions unlike previous redescriptions. WeightSort( ) arranges all weights in sorted order. Corresponding paths of maximum weights are kept first. By giving a threshold value we can filter the remaining paths of lesser weights. Filtrating unwanted paths improves the security between the paths. Because the lesser the weightof( ) the path is, the lesser the relation and so the relation type. On the other side the more the weights are the strong the relation of the path is. Hence the reliability between paths can be measured in positive correlation with the path weight.

I. Experimental Results

For the graph given in Fig. 10, the generated results are displayed in Fig. 11. Source and target nodes are (1,4). Fig. 10 is an undirected graph and with no relations for the edges. Fig. 12 is a directed graph with relations. Fig. 13 shows the results of this graph. These outputs are obtained by executing the Algorithm RED-Depth. Time complexity is directly proportional to the number of nodes in the Graph.

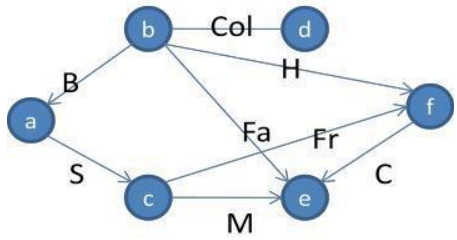


Fig. 12 Directed Graph with relations

The graph in Figure 12 is a directed one with the edges specifying relation type. Full text of relation type is given in Table 1. Path b-e is of the weight 5, path b-a-c-e has the weight 12 (4+4+4), path b-f-e has the weight 8 (5+3), path b-a-c-f-e is of the weight 13(4+4+2+3). Among the derived relational redescriptions, paths of weights 12 and 13 are nearer. That is, paths b-a-c-e and b-a-c-f-e are stronger paths than the remaining ones. Path b-f-c-e is considered with a path weight of 12(5+2+5).

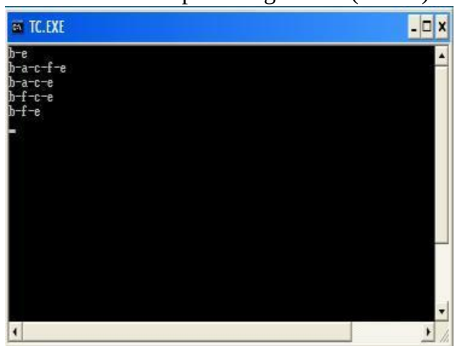


Fig. 13 Unique Paths for the graph of Fig 12

Here in the figure a directional edge is taken but can be considered undirected.

IV. Summary and Conclusions

For the first time the concept of redescribing paths between Social networks is analyzed in this work. RED-Depth Algorithm has found different paths and their corresponding weights between given nodes in a network. This algorithm has also found the patterns of interaction between the nodes of interest. Weightsort algorithm arranges weights of each path in sorted order using Bubble sort. RED-Depth does not take in to account the number of edges between nodes. This can be done by taking the number of hops from the given node to reach the target node. Adding this would be a new parameter to find out the distance between interested nodes. At this point, we

ISSN: 2278 – 1323

have considered only paths emanating and this would be extension to this one. Our approach of finding different paths and their weights along those paths will be a pre-module for the relation type task in Socio-matrices. Relational redescriptions identified as paths between two persons in a Social network are implemented through RED-Depth algorithm. Enhancements of this concept would be a group of people whether related positively or agree or disagree with some other people can be analyzed once they identify paths and their relation strengths.

Also, static network is considered rather than dynamic network in which nodes can connect and disconnected independently. Future enhancements of this work can be extended to find the structure of the paths with specifications mentioned.

References

[1]. Galbrun.E. and Angelika Kimmig (2012), towards finding Relational Redescriptions, *Discovery Science*, Vol: 7569, pp: 52-66.

[2]. Nathalie Henry and Jean-Daniel Fekete (2006), Matrix Explorer: A Dual-Representation System to Explore Social Networks. *IEEE Transactions. Computer Graph.* Vol: 12, Issue:5, pp: 677-684.

[3]. Zeqian Shen and Kwan-Liu Ma(2007), Path Visualization for Adjacency Matrices, Eurographics ,*IEEE-VGTC Symposium* on visualization

[4]. Kate Ehrlich and Inga Carboni(2005), Inside Social Network Analysis, *IBM Technical Report*.pp:5-10.

[5]. Cross.R and Parker.A.(2004), The Hidden Power of Social Networks. *Harvard University Press*.USA.

[6]. Noh.J.D. and Rieger.H.(2004), Random Walks on Complex Networks. *Phys.Rev.Let.92.118701*.

[7]. Cross.R.,Liedtka.J.andWeiss.L.(2005), A Practical Guide To Social Networks. *Harvard Business Review*.

[8]. Stephenson.K.A. and Zelen.M (1989), Rethinking centrality: Methods and examples. *Social Networks.* Vol: 11, pp: 1–37.

ISSN: 2278 – 1323