Computing connection strength in probabilistic graphs

Dmitri V. Kalashnikov Sharad Mehrotra
Computer Science Department
University of California, Irvine
http://www.ics.uci.edu/~dvk/RelDC

Contents

Abstract

Many data mining algorithms view datasets as attributed relation graphs (ARGs) in which nodes correspond to entities and edges to relationships. However, frequently real-world datasets contain uncertainty and can be represented only as *probabilistic ARGs (pARGs)*, i.e. as ARGs where each edge has associated with it a non-zero probability of that edge to exist – *probabilistic edge*. In this paper we first introduce an interesting novel concept of probabilistic ARGs and then present an algorithm for efficient computation of the important *connection strength* metric directly on probabilistic ARGs. Such a metric is useful for analyzing social and collaboration networks, to measure 'impact', and other purposes.

1 Introduction

Attributed relational graphs are used widely by data mining and other communities to visualize relational datasets as entities (nodes) connected via relationships (edges). A sample ARG for the publications domain is shown in Figure 4: it shows authors connected to their publications and also their departments. There is an increasing interest in developing algorithms, measures, and software tools for quantitative analysis of graph data especially as graphs become too large and complex for manual visual analysis.

Connection strength metric. One such important measure that has caught the attention recently is the *connection strength* metric $c(u, v)$. This metric, given two nodes u and v in an ARG, measures the degree of interconnection between these nodes via various u-v paths. Nodes u and v correspond to some real-world entities e_u and e_v, and $c(u, v)$ reflects how strongly e_u and e_v are related to each other via (chains of) relationships. Such a metric is useful for analyzing social and collaboration networks, and other purposes. For example, in a citation network such a metric can be used as one of the methods to compute the 'impact' of a given author on a given area or to compute how closely the research areas of two authors are related to each other. The most prominent work on computing $c(u, v)$ metric is by Faloutsos et al. [8] where various aspect of developing a proper *model* for computing $c(u, v)$ metric are analyzed and methods for scaling the computation of this metric to large ARGs are discussed in great detail. Another related work is [21] in which White et al. present numerous approaches for computing 'relative importance' of certain nodes with respect to the 'root nodes' and use this measure in the context of analyzing social, collaboration and terrorist networks. Computing $c(u, v)$ metric is equivalent to computing the relative importance of node u with respect to one root node v. This metric has also been used in [17] for the purpose of highly accurate data cleaning. The most advanced models for computing $c(u, v)$ in [8, 21, 17] are all closely associated with computing the probability of reaching node v from node u via random walks in the ARG.

Probabilistic ARGs. Previous work on computing $c(u, v)$ metric deals with regular ARGs. ARGs are widely used by various communities (database, data mining, multimedia, bioinformatics, etc.) as a convenient way to represent and visualize datasets. However, many real-world datasets, including the datasets used in [21, 17], contain uncertainty and cannot be fully represented by regular ARGs.

For example, consider a dataset constructed by crawling the web which stores researchers and their affiliations with universities. If the crawler determines with 100% confidence that a given researcher R is affiliated with university U, then the situation is simple and R and U will be connected via the corresponding edge in the ARG. However, the affiliation of R might not be known with 100% confidence. Crawlers (or extraction software) use certain rules to extract information from raw data and they are often confused. The crawler being used to extract affiliations might only be able to derive that with, say, 60% probability R is affiliated with university U_1, and with 40% R is with U_2. This can happen, for instance, because, when the extractor was parsing the raw data (e.g., a web page), it encountered both U_1 and U_2 at the place where it expected to see just one affiliation of R. Or, the description of the affiliation in the raw data was ambiguous such that it could match both U_1 and U_2. The latter situation is very frequent, especially for names: e.g. 'J. Doe' can match both 'John Doe' and 'Jane Doe'. While this uncertainty cannot be captured in a regular ARG, it can be naturally reflected in a *probabilistic ARG* by creating two edges (R, U_1) and (R, U_2) and associating with them probabilities of 60% and 40% to exist.

A probabilistic ARG (pARG) is an ARG that has *probabilistic edges*. A probabilistic edge has associated with it a non-zero probability of that edge to exist. pARGs are a more natural representation of many real-world datasets than ARGs due to the inherent uncertainty in those datasets. In this paper we address the challenge of computing $c(u, v)$ metric directly on pARGs.

While a pARG can be viewed as a regular ARG where each edge has an attribute corresponding to the probability of this edge to exist, we distinguish ARGs and pARGs because of the semantic difference: traditional algorithms that work on regular ARGs assume that all edges do exist, which is not the case for pARGs. Probabilistic edges is not the only difference between ARGs and pARGs: pARGs can also represent one type of 'mutual exclusion' of edges as explained below. Consider the example above again

where edges (R, U_1) and (R, U_2) have probabilities of 60% and 40% associated with them. This fact alone can have two interpretations. The first one is that R is affiliated with only one of U_1 or U_2, so if we assume at some point that R_1 is affiliated with U_1, then we must assume edge (R, U_2) is absent. The second interpretation is that R can be affiliated with both U_1 and U_2 independently, so if we again assume at some point that R is with U_1, it does not mean that edge (R, U_2) is absent.

To avoid this ambiguity, the two cases are represented differently in pARGs. The simple case where R can be independently affiliated with both U_1 and U_2 is shown in Figure 1(b). The more interesting case where R can be affiliated with only one, U_1 or U_2, is shown in Figure 1(c). In this figure R is shown to be connected to a special *choice* node v^*_R via an edge labeled with probability 1. The choice node represents mutual exclusion and is connected to U_1 and U_2 via edges labeled with probabilities 0.6 and 0.4. (Note, in the case of mutual exclusion probabilities should add up to 1 whereas in general its is not so for the independent case.)

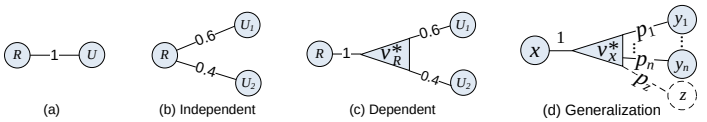

Figure 1: Representing probabilistic edges

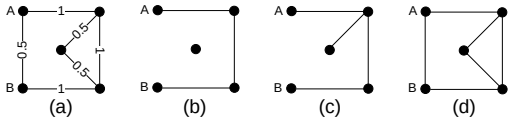

Figure 2: pARG maps to several ARGs

It is possible to create a single generic representation for a probabilistic edge that would cover mutual exclusion and independence cases, examples of which are shown in Figure 1(b) and 1(c). This generic representation is illustrated in Figure 1(d) where node x is shown to be connected to nodes $y_1, y_2, \ldots, y_n, z$ $(n \geq 1)$ via a choice node v^*_x, and where $p_1 + p_2 + \ldots + p_n + p_z = 1$. This representation semantically denotes that there are $(n+1)$ mutually exclusive possibilities: either edge (x, p_i) $(i = 1, 2, \ldots, n)$ exists with probability p_i or none of those edges edges exist with probability p_z. The latter case is represented via a *virtual* edge (x, z) – this edge is not added to the ARG. Nodes $y_1, y_2, \ldots, y_n$ are called the *options* of choice node v^*_x, edges (v^*_x, y_i) $(i = 1, 2, \ldots, n)$ are called the *option-edges* of v^*_x. Using this generic representation it is straightforward to represent the case in Figure 1(c). An independent edge (R, U_1) in Figure 1(b) can be represented as $R \rightarrow v^*_R \rightarrow U_1$, that is R connected to a choice node v^*_R and which is connected to U_1 via an edge labeled with probability 0.6. Edge (R, U_2) will have a similar representation: $R \rightarrow v^*_{R2} \rightarrow U_1$. Notice, v^*_R and v^*_{R2} are just a notation for two different choice nodes, we could have named them differently.

Definition 1. A probabilistic edge is an edge like shown in Figure 1(d).

Definition 2. A probabilistic ARG is an ARG with probabilistic edges.

In the subsequent discussion we will not use the long representation for independent edges (as in $R \rightarrow v^*_R \rightarrow U_1$), rather representing them by a single edge (as (R, U_1) in Figure 1(b)).

Figure 2 illustrates another interesting property of pARGs: each pARG maps on to a family of regular ARGs. Figure 2(a) shows one pARG where three edges are labeled with probability of 0.5 and each edge exists independently from other edges – that is, there is no mutual exclusion among them. This pARG maps on to 2^3 regular ARGs. For instance, if we assume that none of the three edges is present (the probability of which is 0.5^3) then the pARG will be instantiated to the ARG in Figure 2(b). Figures 2(c) and 2(d) show other two possible instantiations of it, each having the same probability of occurring of 0.5^3.

pARGs are useful because many datasets contain uncertainty, removing which might be a hard or impossible task. If we could remove this uncertainty, we could have replaced all probabilistic edges with regular edges. In other words, we could have instantiated a given pARG to the only *corresponding correct regular ARG*. Ideally we would have liked to compute all the statistics on this correct ARG, but this ARG is unknown and often infeasible to derive. Given the above, if our goal is to compute some statistic on a dataset represented by the corresponding pARG, the natural question is whether this statistic can still be accurately computed (or, at least well approximated) by choosing one instantiation of pARG and computing it on this instantiation. This solution is known not to work correctly in general case. For instance, consider computing the shortest path between nodes A and B for the pARG in Figure 2(a). If the correct instantiation of this pARG is the ARG in Figure 2(b) but we choose the ARG from Figure 2(d) to

compute the shortest path, then we will compute it as 1 instead of 3. While the above is just an artificial example, there are real-world situations where similar solutions have led to errors in computations, as explained next.

CiteSeer	Cleaned CiteSeer
1. D. Johnson	1. John Ousterhout (Pacbell)
2. J. Ullman	2. Willy Zwaenepoel (Rice)
3. A. Gupta	3. David Culler (Berkeley)
4. R. Milner	4. Jennifer Widom (Stanford)
5. R. Rivest	5. Thorsten Von Eicken
6. S. Shenker	6. Klaus Schauser (UCSB)
7. V. Jacobson	7. Serge Abiteboul (Inria)
8. M. Garey	8. Bart Selman (Cornell)
9. R. Tarjan	9. Rakesh Agrawal (IBM)
10. S. Floyd	10. Luca Cardelli

Figure 3: Wrong statistics on not-cleaned data.

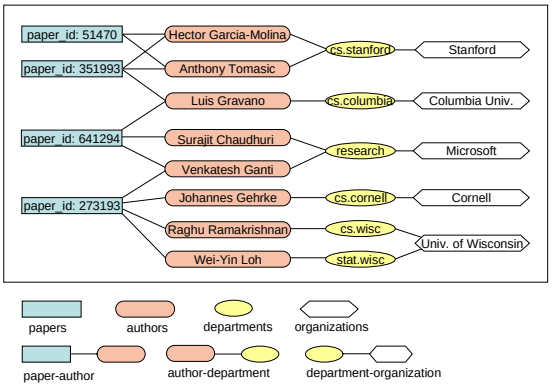

Figure 4: ARG for a subset of CiteSeer

Figure 3 shows the top-10 most cited authors based on CiteSeer data. CiteSeer can be viewed as a database of research publications collected by crawling the web, most of which are for the period of 1992–now. CiteSeer can be accessed via the web and provides multiple statistics on the publication data, in particular information about the most cited authors is available. Due to the popularity of CiteSeer, the latter information is frequently used in CVs, and as one of the tools when university make important decisions, such as hiring faculty and giving tenure.

Unfortunately, there are many known problems with this dataset, one of them being that the statistics are computed on uncleaned data. Potentially, CiteSeer can be represented as a regular ARG with nodes for authors and publications, and edges for citations and authorship. However deriving the correct version of this ARG is difficult. For example, it is difficult to link authors and papers in the ARG because of the ambiguity in author names: e.g, author name 'J. Smith' in publication P can match authors A_1:'John Smith' and A_2:'Jane Smith' and thus it is a challenge to determine which edge, $E_1 = (P, A_1)$ or $E_2 = (P, A_2)$, should be created. From what is known, CiteSeer would resolve this situation by creating *both* edges, E_1 and E_2, thus creating a wrong ARG.

There are algorithms that allow to clean data and compute a more accurate ARG for CiteSeer. The left column of Figure 3 shows the most cited authors as reported by CiteSeer, and the right column shows the same statistic on a cleaner version of the same data. The top-10 lists are completely different and in general top-k lists are different as well. There are many other factors corroborating that the top-k list computed on the correct ARG for CiteSeer is different from what is currently reported by CiteSeer. Notice, we would caution to avoid using both of the top-10 list from Figure 3 as being the correct representation of the top-10 most cited authors due to other issues with CiteSeer dataset and the cleaning algorithm.

The example above shows a real-world situation where statistics are computed inaccurately and one of the reasons for that is computing those statistics (semantically) on a wrong regular ARG. CiteSeer can be represented as a pARG and this wrong regular ARG can be viewed as a (wrong) instantiation of the pARG for CiteSeer. Instead of using the wrong ARG to compute statistics, either the correct (or more accurate) instantiation of the pARG should have been computed first, or, alternatively, more accurate statistics could have potentially been computed directly on the pARG itself.

The example above is also interesting because it shows the importance of the algorithm for computing $c(u, v)$ metric presented in Section 3. The cleaner version of the publication data shown in Figure 3 has been obtain by applying a data cleaning algorithm [17] which relies on using this very metric and its ability to work directly with pARGs.

The example in Figure 3 underscores the significance of the novel concept of pARGs, introduced in this paper, and, consequently, the need to adjust the traditional algorithms that work with ARGs to work with pARGs as well. The challenge in designing those algorithms comes from the fact that if a

pARG has n independent edges that are labeled with non-1 probabilities, then this pARG maps into the exponential number (i.e., 2^n) of regular ARGs, where the probability of each instantiation is determined by the probability of the corresponding combination of edges to exist. Algorithms that work with pARGs should be able to account for the fact that some of the edges exist only with certain probabilities. If such an algorithm computes a certain measure on the pARG it should avoid computing it naïvly by computing it on each of 2^n instantiations of the pARG separately and then outputting the probabilistic average as the answer. Instead smart techniques should be designed capable of computing the same answer by applying more efficient methods.

The primary contributions of this paper are:

1. Introduction of the notion of probabilistic ARGs as a convenient method for representing many real-world datasets.
2. The main contribution is the algorithm for computing $c(u, v)$ metric on probabilistic ARGs.
3. Finally, we believe this paper opens up a plethora of opportunities for other researchers to devise algorithms that work directly with probabilistic ARGs .

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 develops the algorithm for computing $c(u, v)$ metric on pARGs. Section 4 summarizes the results of extensive experimental evaluation of the $c(u, v)$ metric. Finally, Section 5 concludes the paper.

2 Related Work

In this paper we introduce the notion of a probabilistic ARG (pARG) as a convenient way to represent many real-world datasets that contain uncertainty [17]. Such a representation can be, for example, naturally derived by an extraction software that can build a pARG from raw data. The probabilities on edges can be then further refined by applying domain knowledge.

Even though there is an abundance of general-purpose graph theoretic techniques [6] and in particular algorithms that work with ARGs, we are unaware of any approach that work with probabilistic ARGs as we defined them. In this paper we present an algorithm that computes the connection strength measure between any given two nodes in a pARG. Similar algorithm (on regular ARGs) are discussed in [21, 8] and in the social network analysis literature [20]. The connection strength measure is related to ranking nodes in graphs. The two most distinguished contribution to ranking nodes in Web graphs are the PageRank algorithm [3] by Brin et al. and HITS algorithm [14] by Kleinberg and there have been many extensions of these two algorithms [1, 15]. All these algorithms however compute 'global' ranking of nodes whereas connection strength is more related to personalized versions of PageRank and HITS, such as proposed in [5, 12, 10], capable of biasing the result. Other related work include graph clustering, partitioning algorithms and random walks in graphs [16, 13, 19, 4, 7, 2, 11, 18].

3 Probabilistic model for computing connection strength

In this section we study a connection strength model, called the *probabilistic model (PM)* for computing $c(u, v)$ metric in pARGs. A connection strength metric allows to measure how strongly two entities are related to each other via (chains of) relationships that exist between them. For instance, if a researcher R is affiliated with a university U_1 and also collaborates with some researchers from university U_2, then there is a certain degree of connection between R and U_1 and there is a weaker connection between R and U_2. Such a measure is used for analysis of social, collaboration and terrorist networks and other purposes.

A natural way to compute the connection strength $c(u, v)$ between node u and v in graph G is to compute it as the probability to reach node v from node u via random walks in G. Most advanced models for computing $c(u, v)$ [17, 8, 21] are all closely related to this method, Appendix A.2 motivates this choice.

In PM the connection strength between nodes u and v is computed as the sum of the connection strengths of all simple paths between u and v: $c(u, v) = \sum_{\mathscr{P} \in \mathcal{P}_L(u,v)} c(\mathscr{P})$. This formula iterates over the

set $\mathcal{P}_L(u,v)$ of all L-short simple paths that exist between nodes u and v, where a path is L-short if its length does not exceed L and a path is simple if it does not contain duplicate nodes. Notation $c(\mathscr{P})$ in the above formula denotes the connection strength of path $\mathscr{P}$ which is computed as the probability to follow this path in graph G. Such computed $c(u,v)$ corresponds to the probability of reaching node v from node u via L-short simple paths. Most of the discussion in the rest of this section will be devoted to computing the connection strength $c(\mathscr{P})$ of a single path $\mathscr{P}$. Appendix A.5 justifies why it is possible to simply sum up such individually computed connection strength $c(\mathscr{P})$ of each path $\mathscr{P} \in \mathcal{P}_L(u,v)$ to obtain the desired total probability to reach node v from node u.

Notation	Meaning
$x^{\exists}$	event "x exists" for (edge,path) x
$x^{\nexists}$	event "x does not exist"
$x^{\rightarrow}$	event corresponding to following x
$dep(e_1,e_2)$	if events e_1 and e_2 are dependent, then $dep(e_1,e_2) = \texttt{true}$, else $\texttt{false}$
$\mathrm{P}(x^{\exists})$	probability that (edge) x exists
$\mathrm{P}(x^{\rightarrow})$	probability to follow ('go via') x
$\mathscr{P}$	the path being considered
v_i	i-th node on path $\mathscr{P}$
E_i	(v_i, v_{i+1}) edge on path $\mathscr{P}$
E_{ij}	edge labeled with probability p_{ij}

Notation	Meaning
a_{ij}	$a_{ij} = 1$ if edge E_{ij} exists; else $a_{ij} = 0$
$a_{i0} = 1$	dummy variables: $a_{i0} = 1$ (for all i)
$p_{i0} = 1$	dummy variables: $p_{i0} = 1$ (for all i)
$opt(E)$	if edge E is an option-edge, then $opt(E) = \texttt{true}$, else $opt(E) = \texttt{false}$
v^*_E	if edge E is an option-edge, then v^*_E is the choice node associated with E
$\mathbf{a}$ (vec)	vector, $\mathbf{a} = (a_{10}, a_{11}, \ldots, a_{(k-1)n_{k-1}})$
$\mathbf{a}$ (set)	$\mathbf{a} = \{a_{ij} :$ for all $i,j\}$
$\mathbf{a}$ (var)	at each moment variable $\mathbf{a}$ is one instantiation of $\mathbf{a}$ as a vector

Table 1: Probabilistic model: Terminology (split into two parts)

3.1 Preliminaries

In Section 1 we have introduced the notion of a *choice* node, see Figure 1(d). We will consider pARGs with independent probabilistic edges and with mutual exclusion where $p_1 + p_2 + \ldots + p_n = 1$, e.i. p_z is always 0.[1]

Notation. We will compute probabilities of certain events. Notation $\mathrm{P}(A)$ refers to the probability of event A to occur. We use $E^{\exists}$ to denote event "E exists" for edge E. Similarly, we use $E^{\nexists}$ for event "E does not exist". So, $\mathrm{P}(E^{\exists})$ refers to the probability that E exists. We will consider situations where the algorithm computes the probability to follow (or 'go via') a specific edge E, usually in the context of a specific path. This probability is denoted as $\mathrm{P}(E^{\rightarrow})$. We will use $dep(e_1,e_2)$ notation as follows: $dep(e_1,e_2) = \texttt{true}$ if and only if events e_1 and e_2 are dependent. Notation $\mathscr{P}$ denote the path being currently considered. Table 1 summarizes the notation.

Example 3.1.1 (Basic case: PM on a regular ARG). Before we discuss how PM computes $c(\mathscr{P})$ on pARGs, let us first consider how it achieves that on a regular ARG – without probabilistic edges. Consider computing $c(\mathscr{P})$ for path $\mathscr{P} = A \rightarrow B \rightarrow 2$ in Figure 5. For this simplest case PM computes the probability to follow path $\mathscr{P}$ as the product of probabilities to follow edges AB and $B2$, which are computed as follows. For $\mathscr{P}$, we start from node A and it is possible to follow only edge AB, so the probability to follow AB is 1. Then from node B it is possible to follow edges $B1, B2, \ldots, BN$, so the probability to follow specifically $B2$ edge is computed as $\frac{1}{N}$ and the total probability to follow $\mathscr{P}$ is therefore $1 \cdot \frac{1}{N} = \frac{1}{N}$. □

When computing $c(u,v)$ on a probabilistic ARG the key is to avoid using the naïve procedure, which computes $c(u,v)$ by instantiating the ARG to regular ARGs, computing $c(u,v)$ on these ARGs and then outputting probabilistic average. The rest of Section 3 discusses a method for computing $c(u,v)$ more efficiently.

[1] We expect most of pARGs to fall into this category.

Toy examples: PM on probabilistic ARGs. Let us introduce PM on pARGs by analyzing two toy examples shown in Figures 6 and 7. Each figure shows a small graph in which path $\mathscr{P} = A \to B \to C \to D \to E$ will be of interest to us.

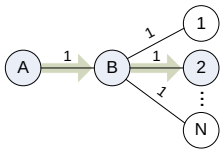

Figure 5:

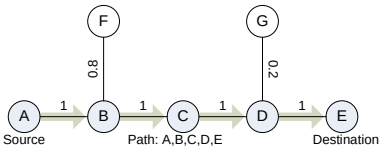

Figure 6: Toy example: independent case

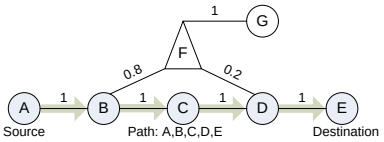

Figure 7: Toy example: dependent case

In Figure 6 we assume the events "edge BF is present" and "edge DG is present" are *independent*. The probability of the event "edge BF is present" is 0.8. The probability of the event "edge DG is present" is 0.2. In Figure 7 node F represents a choice node and BF and DF are its option-edges. Events "edge BF exists" and "edge DF exists" are mutually exclusive (and hence strongly *dependent*): if one edge is present the other edge must be absent due to the semantics of the choice node.

PM computes the connection strength $c(\mathscr{P})$ of path $\mathscr{P}$ as the probability to follow path $\mathscr{P}$: $c(\mathscr{P}) = \mathrm{P}(\mathscr{P}^{\to})$. In PM computing $c(\mathscr{P})$ is a two step process. PM first computes the probability $\mathrm{P}(\mathscr{P}^{\exists})$ that path $\mathscr{P}$ exists, then it computes the probability $\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists})$ to follow $\mathscr{P}$ given that $\mathscr{P}$ exists. Then PM computes $c(\mathscr{P})$ as $c(\mathscr{P}) = \mathrm{P}(\mathscr{P}^{\to}) = \mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists})\mathrm{P}(\mathscr{P}^{\exists})$.

Thus the first step is to compute $\mathrm{P}(\mathscr{P}^{\exists})$. A path exists if each edge on that path exists. For the path $\mathscr{P}$ in Figures 6 and 7, probability $\mathrm{P}(\mathscr{P}^{\exists})$ is equal to $\mathrm{P}(AB^{\exists} \cap BC^{\exists} \cap CD^{\exists} \cap DE^{\exists})$. If the existence of each edge in the path is independent from the existence of other edges, e.g. like for the cases shown in Figures 6 and 7, then $\mathrm{P}(\mathscr{P}^{\exists}) = \mathrm{P}(AB^{\exists} \cap BC^{\exists} \cap CD^{\exists} \cap DE^{\exists}) = \mathrm{P}(AB^{\exists})\mathrm{P}(BC^{\exists})\mathrm{P}(CD^{\exists})\mathrm{P}(DE^{\exists}) = 1$.

The second step is to compute the probability $\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists})$ to follow path $\mathscr{P}$, given that $\mathscr{P}$ exists. Once this probability is computed, we can compute $c(p)$ as $c(\mathscr{P}) = \mathrm{P}(\mathscr{P}^{\to}) = \mathrm{P}(\mathscr{P}^{\exists})\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists})$. The probability $\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists})$ is computed differently for the cases in Figures 6 and 7. This will lead to different values of $c(\mathscr{P})$.

Example 3.1.2 (Independent edge existence). Let us first consider the case where the existence of each edge is independent from the existence of the other edges. In Figure 6 two events "BF exists" and "DG exists" are independent. The probability to follow path $\mathscr{P}$ is the product of probabilities to follow each of the edges on the path: $\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists}) = \mathrm{P}(AB^{\to}|\mathscr{P}^{\exists})\mathrm{P}(BC^{\to}|\mathscr{P}^{\exists})\mathrm{P}(CD^{\to}|\mathscr{P}^{\exists})\mathrm{P}(DE^{\to}|\mathscr{P}^{\exists})$. Given path $\mathscr{P}$ exists, the probability to follow edge AB in path $\mathscr{P}$ is one. The probability to follow edge BC is computed as follows. With probability 0.2 edge BF is absent, in which case the probability to follow BC is 1. With probability 0.8 edge BF is present, in which case the probability to follow BC is $\frac{1}{2}$ – because there are two links, BF and BC, that can be followed. Thus the total probability to follow BC is $0.2 \cdot 1 + 0.8 \cdot \frac{1}{2} = 0.6$. Similarly, the probability to follow CD is 1 and the probability to follow DE is $0.8 \cdot 1 + 0.2 \cdot \frac{1}{2} = 0.9$. The probability to follow path $\mathscr{P}$, given it exists, is the product of probabilities to follow each edge of the path which is equal to $1 \cdot 0.6 \cdot 1 \cdot 0.9 = 0.54$. Since for the case shown in Figure 6 path $\mathscr{P}$ exists with probability 1, the final probability to follow $\mathscr{P}$ is $c(\mathscr{P}) = \mathrm{P}(\mathscr{P}^{\to}) = 0.54$. □

Example 3.1.3 (Dependent edge existence). Let us now consider the case where the existence of an edge can depend on the existence of the other edges. For the case shown in Figure 7 edges BF and DF cannot exist both at the same time. To compute $\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists})$ we will consider two cases separately: $BF^{\exists}$ and $BF^{\not\exists}$. That way we will be able to compute $\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists})$ as $\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists}) = \mathrm{P}(BF^{\exists}|\mathscr{P}^{\exists})\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists} \cap BF^{\exists}) + \mathrm{P}(BF^{\not\exists}|\mathscr{P}^{\exists})\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists} \cap BF^{\not\exists})$.

Let us first assume $BF^{\exists}$ (i.e., edge BF is present) and then compute $\mathrm{P}(BF^{\exists}|\mathscr{P}^{\exists})\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists} \cap BF^{\exists})$. For the case of Figure 7, if no assumptions about the presence or absence of DF have been made yet, $\mathrm{P}(BF^{\exists}|\mathscr{P}^{\exists})$ is simply equal to $\mathrm{P}(BF^{\exists})$ which is equal to 0.8. If BF is present then DF is absent and the probability to follow $\mathscr{P}$ is $\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists} \cap BF^{\exists}) = 1 \cdot \frac{1}{2} \cdot 1 \cdot 1 = \frac{1}{2}$. Now let us consider the second case $BF^{\not\exists}$ (and thus $DF^{\exists}$). The probability $\mathrm{P}(BF^{\not\exists}|\mathscr{P}^{\exists})$ is 0.2. For that case $\mathrm{P}(\mathscr{P}^{\to}|\mathscr{P}^{\exists} \cap BF^{\not\exists})$ is equal to

$1 \cdot 1 \cdot 1 \cdot \frac{1}{2} = \frac{1}{2}$. Thus $\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists}) = 0.8 \cdot \frac{1}{2} + 0.2 \cdot \frac{1}{2} = 0.5$. So $c(\mathscr{P}) = \mathrm{P}(\mathscr{P}^{\rightarrow}) = 0.50$, which is different from that of the previous experiment. □

3.2 Independent edge existence

Let us consider how to compute path connection strength in general case, assuming the existence of each edge is independent from existence of the other edges.

3.2.1 General formulae

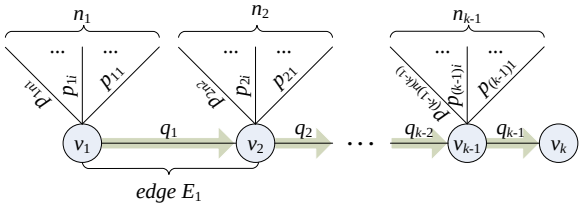

Figure 8: Independent case. Computing $c(\mathscr{P})$ for $\mathscr{P} = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$. Edges that are not possible to follow w.r.t. $\mathscr{P}$ are not shown.

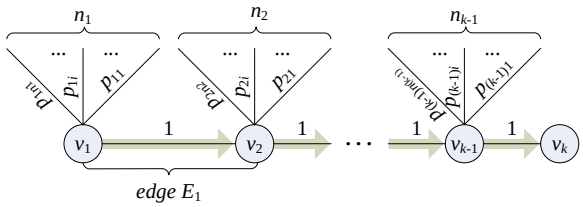

Figure 9: An equivalent of Figure 8 with an additional assumption that path $\mathscr{P}$ exists.

In general, any path $\mathscr{P}$ can be represented as a sequence of k nodes $\langle v_1, v_2, \ldots, v_k \rangle$ or as a sequence of $(k-1)$ edges $\langle E_1, E_2, \ldots, E_{(k-1)} \rangle$, as illustrated in Figure 8, where $E_i = (v_i, v_{i+1})$ and $\mathrm{P}(E_i^{\exists}) = q_i$ $(i = 1, 2, \ldots, k-1)$. We will refer to edges labeled with probabilities p_{ij} (for all i, j) in this figure as E_{ij}. The goal is to compute the probability to follow path $\mathscr{P}$, which is the measure of the connection strength of path $\mathscr{P}$:

$$c(\mathscr{P}) = \mathrm{P}(\mathscr{P}^{\rightarrow}) = \mathrm{P}(\mathscr{P}^{\exists})\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists}). \tag{1}$$

The probability that $\mathscr{P}$ exists is equivalent to the probability that each of its edges exists:

$$\mathrm{P}(\mathscr{P}^{\exists}) = \mathrm{P}(\bigcap_{i=1}^{k-1} E_i^{\exists}). \tag{2}$$

Given our assumption of the independence, $\mathrm{P}(\mathscr{P}^{\exists})$ can be computed as

$$\mathrm{P}(\mathscr{P}^{\exists}) = \prod_{i=1}^{k-1} \mathrm{P}(E_i^{\exists}) = \prod_{i=1}^{k-1} q_i. \tag{3}$$

To compute $\mathrm{P}(\mathscr{P}^{\rightarrow})$ we now need to compute $\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists})$. In turn, to compute $\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists})$ let us analyze how labels p_{ij} and q_i (for all i, j) in Figure 8 will change if we assume that $\mathscr{P}$ exists. We will compute the corresponding new labels, $\widetilde{p}_{ij}$ and $\widetilde{q}_i$, and reflect the changes in Figure 9. Since q_i is defined as $q_i = \mathrm{P}(E_i^{\exists})$ and p_{ij} is defined as $p_{ij} = \mathrm{P}(E_{ij}^{\exists})$, the new labels are computed as $\widetilde{q}_i = \mathrm{P}(E_i^{\exists}|\mathscr{P}^{\exists}) = 1$ and $\widetilde{p}_{ij} = \mathrm{P}(E_{ij}^{\exists}|\mathscr{P}^{\exists})$. Given our assumption of independence, $\widetilde{p}_{ij} = p_{ij}$. The new labeling is shown in Figure 9.

Let us define a variable a_{ij} for each edge E_{ij} (labeled p_{ij}) as follows: $a_{ij} = 1$ if and only if edge E_{ij} exists; otherwise $a_{ij} = 0$. Also, for notational convenience, let us define two sets of dummy variables a_{i0} and p_{i0}: $a_{i0} = 1$ and $p_{i0} = 1$ $(i = 1, 2, \ldots, k-1)$.[2] Let $\mathbf{a}$ denote a vector consisting of all a_{ij}'s: $\mathbf{a} = (a_{10}, a_{11}, \ldots, a_{(k-1)n_{k-1}})$. Let $\mathcal{A}$ denote the set of all possible instantiations of $\mathbf{a}$, i.e. $|\mathcal{A}| = 2^{n_1+n_2+\cdots+n_{k-1}}$. Then probability $\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists})$ can be computed as

$$\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists}) = \sum_{\mathbf{a} \in \mathcal{A}} \left\{ \mathrm{P}(\mathscr{P}^{\rightarrow}|\mathbf{a} \cap \mathscr{P}^{\exists}) \mathrm{P}(\mathbf{a}|\mathscr{P}^{\exists}) \right\}, \tag{4}$$

[2] Intuitively (1) $a_{i0} = 1$ corresponds to the fact that edge E_i exists given path $\mathscr{P}$ exists; and (2) $p_{i0} = 1$ corresponds to $p_{i0} = \mathrm{P}(E_i^{\exists}|\mathscr{P}^{\exists}) = 1$.

where $\mathrm{P}(\mathbf{a}|\mathscr{P}^{\exists})$ is the probability of instantiation $\mathbf{a}$ to occur while assuming $\mathscr{P}^{\exists}$. Given our assumption of independence of probabilities, $\mathrm{P}(\mathbf{a}|\mathscr{P}^{\exists}) = \mathrm{P}(\mathbf{a})$. Probability $\mathrm{P}(\mathbf{a})$ can be computed as

$$\mathrm{P}(\mathbf{a}|\mathscr{P}^{\exists}) = \mathrm{P}(\mathbf{a}) = \prod_{\substack{i=1,2,\ldots,k-1 \\ j=0,1,\ldots,n_i}} p_{ij}^{a_{ij}}(1-p_{ij})^{1-a_{ij}}. \tag{5}$$

Probability $\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathbf{a} \cap \mathscr{P}^{\exists})$, which is the probability to go via $\mathscr{P}$ given (1) a particular instantiation of $\mathbf{a}$; and (2) the fact that $\mathscr{P}$ exists, can be computed as

$$\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathbf{a} \cap \mathscr{P}^{\exists}) = \prod_{i=1}^{k-1} \frac{1}{1+\sum_{j=1}^{n_i} a_{ij}} \equiv \prod_{i=1}^{k-1} \frac{1}{\sum_{j=0}^{n_i} a_{ij}}. \tag{6}$$

Thus

$$\mathrm{P}(\mathscr{P}^{\rightarrow}) = \left(\prod_{i=1}^{k-1} q_i\right)\left(\sum_{\mathbf{a}\in\mathcal{A}}\left\{\left[\prod_{i=1}^{k-1}\frac{1}{\sum_{j=0}^{n_i} a_{ij}}\right]\left[\prod_{ij} p_{ij}^{a_{ij}}(1-p_{ij})^{1-a_{ij}}\right]\right\}\right). \tag{7}$$

3.2.2 Computing path connection strength in practice

Notice, Equation (7) iterates through all possible instantiations of $\mathbf{a}$ which is impossible to compute in practice given $|\mathcal{A}| = 2^{n_1+n_2+\cdots+n_{k-1}}$. This equation must be simplified to make the computation feasible.

Computing $\mathbf{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists})$ as $\prod_{i=1}^{k-1}\mathbf{P}(E_i^{\rightarrow}|\mathscr{P}^{\exists})$. To achieve the simplification, we will use our assumption of independence of probabilities which allows us to compute $\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists})$ as the product of the probabilities to follow each individual edge in the path:

$$\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists}) = \prod_{i=1}^{k-1} \mathrm{P}(E_i^{\rightarrow}|\mathscr{P}^{\exists}). \tag{8}$$

Let $\mathbf{a}_i$ denote vector $(a_{i0}, a_{i1}, \ldots, a_{in_i})$, that is $\mathbf{a} = (\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_{k-1}})$. Let $\mathcal{A}_i$ denote all possible instantiations of $\mathbf{a}_i$. That is, $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_{k-1}$ and $|\mathcal{A}_i| = 2^{n_i}$. Then

$$\mathrm{P}(E_i^{\rightarrow}|\mathscr{P}^{\exists}) = \sum_{\mathbf{a}_i\in\mathcal{A}_i}\left\{\left[\frac{1}{\sum_{j=0}^{n_i} a_{ij}}\right]\left[\prod_{j=0}^{n_i} p_{ij}^{a_{ij}}(1-p_{ij})^{1-a_{ij}}\right]\right\}. \tag{9}$$

Combining Equations (1), (8) and (9) we have

$$\mathrm{P}(\mathscr{P}^{\rightarrow}) = \left(\prod_{i=1}^{k-1} q_i\right)\prod_{i=1}^{k-1}\left(\sum_{\mathbf{a}_i\in\mathcal{A}_i}\left\{\left[\frac{1}{\sum_{j=0}^{n_i} a_{ij}}\right]\left[\prod_{j=0}^{n_i} p_{ij}^{a_{ij}}(1-p_{ij})^{1-a_{ij}}\right]\right\}\right). \tag{10}$$

The effect of transformation. Notice, using Equation (7) the algorithm will need to perform $|\mathcal{A}| = 2^{n_1+n_2+\cdots+n_{k-1}}$ iterations – one per each instantiation of $\mathbf{a}$. Using Equation (10) the algorithm will need to perform $|\mathcal{A}_1| + |\mathcal{A}_2| + \cdots + |\mathcal{A}_{k-1}| = 2^{n_1} + 2^{n_2} + \cdots + 2^{n_{k-1}}$ iterations. Furthermore, each iteration requires less computation. These factors lead to a significant improvement. This formula can be improved much further as discussed in Appendix A.3.

3.3 Dependent edge existence

In this section we discuss how to compute connection strength if occurrence of edges is not independent. In our model, dependence between two edges arises only when those two edges are option-edges of the same choice node. We next show how to compute $\mathrm{P}(\mathscr{P}^{\rightarrow})$ for those cases.

There are two principal situations we need to address. The first is to handle all choice nodes on the path. The second step is to handle all choice nodes such that a choice node itself is not on the path but at least two of its option nodes are on the path. Next we address those two cases.

3.3.1 Choice nodes on the path

The first case of how to deal with choice nodes on the path is a simple one. There are two sub-cases in this case illustrated in Figures 10 and 18.

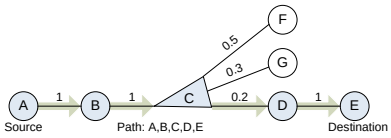

Figure 10: Choice node on the path

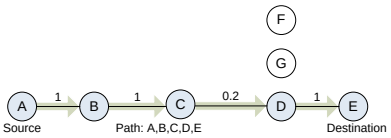

Figure 11: Choice node on the path: removing choice

Figure 10 shows a choice node C on the path which has options D, G, and F. Recall, we compute $\mathrm{P}(\mathscr{P}^{\rightarrow}) = \mathrm{P}(\mathscr{P}^{\exists})\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists})$. When we compute $\mathrm{P}(\mathscr{P}^{\exists})$ each edge of path $\mathscr{P}$ should exist. Thus edge CD must exist, which means edges CG and CF do not exist. Notice, this case is equivalent to the case shown in Figure 11 where (a) edges CG and CF are not there (permanently eliminated from consideration); and (b) node C is just a regular (not a choice) node connected to D via an edge (in this case the edge is labeled 0.2). If we now consider this equivalent case, then we can simply apply Equation (10) to compute the connection strength.

In general, all choice nodes on the path, can be "eliminated" from the path one by one (or, rather, "replaced with regular nodes") using the procedure above.

Figure 18 shows a choice node C on the path which have options B, F, and D, such that $B \rightarrow C \rightarrow D$ is a part of the path $\mathscr{P}$. Semantically, edges CB, CF, and CD are mutually exclusive, so path $\mathscr{P}$ cannot exist. Such paths are said to be *illegal* and they are ignored by the algorithm.

3.3.2 Options of the same choice node on the path

Assume now we have applied the procedure from Section 3.3.1 and all choice nodes are "eliminated" from path $\mathscr{P}$. At this point the probability $\mathrm{P}(\mathscr{P}^{\exists})$ can be computed as $\mathrm{P}(\mathscr{P}^{\exists}) = \prod_{i=1}^{k-1} q_i$. The only case that is left to be considered is where a choice node itself is not on the path but at least two of its options are on the path. An example of such a case is illustrated in Figure 19 where choice node F has four options: G, B, D, and H, two of which B and D belong to the path being considered. After choice nodes are eliminated from the path, the goal becomes to create a formula similar to Equation (10), but for the general "dependent" case. This case it the most complex, but also was found to be very rare in practice, the solution for it is provided in Appendix 3.3.2.

4 Summary of Experimental Evaluation

The algorithm for computing $c(u, v)$ metric proposed in Section 3 has been extensively evaluated in [17] where this metric has been used as the key component of a data cleaning algorithm. The principal difference of that data cleaning algorithm from traditional techniques is that it uses this metric to improve the quality of the results. The approach has been studied on two real (*publications* and *movies*) and synthetic datasets of various sizes. In has been experimentally demonstrated in [17] that combining traditional techniques with a technique that uses $c(u, v)$ metric leads to a very significant improvement in quality of data cleaning as illustrated by a few figures in Appendix A.1.

5 Conclusion

In this paper we have introduced the notion of probabilistic ARGs as a convenient method for representing many real-world datasets and shown how an important metric for measuring connection strength can be implemented directly on pARGs. We believe this publication lays out a foundation for a novel research direction of adapting graph theoretic techniques to work directly on pARGs. As our future work we plan

to address this challenge. Specifically, since many graph theoretic algorithms, such as clustering, rely on some sort of a distance function for their computations, one of the possible ways to achieve the goal is to specify how various kinds of distances can be computed on pARGs.

References

[1] A. Borodin, G. Roberts, J. Rosenthal, and P. Tsparas. Finding authorities and hubs from the link structures of the world wide web. In *Proc. of WWW Conf.*, 2001.
[2] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *ESA Symp.*, 2003.
[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc WWW Conf*, 1998.
[4] H. Bunke and et al. Graph clustering using the weighted minimum common supergraph. In *GbRPR*, 2003.
[5] H. Chang, D. Cohn, and A. McCallum. Creating customized authority lists. In *Proc of Conf on ML*, 2000.
[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms.* MIT Press, 2001.
[7] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *Proc. of SIGKDD Conf.*, 2003.
[8] C. Faloutsos, K. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *SIGKDD*, 2004.
[9] G. Grimmett and D. Stirzaker. *Probability and random processes.* OXFORD University Press, 2002.
[10] T. Haveliwala. Topic-sensitive pagerank. In *Proc. of WWW Conf.*, 2002.
[11] Y.-W. Huang, N. Jing, and E. A. Rundensteiner. Effective graph clustering for path queries in digital map databases. In *Proc. of Int'l Conf. on Information and Knowledge Management*, 1996.
[12] G. Jeh and J. Widom. Scaling personalized web search. In *Proc. of WWW Conf.*, 2003.
[13] G. Karypis and Kumar. Parallel multilevel k-way partitioning for irregular graphs. *SIAM Review*, 41(2), 1999.
[14] Kleinberg. Authoritive sources in a hyperlinked environment. *Journal of*, 46(5), 1999.
[15] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *Proc. of WWW Conf.*, 2000.
[16] B. Luo, R. C. Wilson, and E. R. Hancock. Spectral clustering of graphs. In *GbRPR*, pages 190–201, 2003.
[17] S. Mehrotra, D. Kalashnikov, and Z. Chen. Exploiting relationships for domain-independent data cleaning. *In Submission,* 2004. An extended version is at `www.ics.uci.edu/~dvk/RelDC/TR/TR-RESCUE-04-20.pdf`.
[18] Schenker and et al. Comparison of distance measures for graph-based clustering of documents. In *GbRPR*, 2003.
[19] S. Virtanen. Clustering the Chilean Web. In *Proc. of Latin American Web Congress, IEEE CS*, 2003.
[20] S. Wasserman and K. Faust. *Social Network Analysis.* Cambridge University Press, 1999.
[21] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *Proc. of SIGKDD*, 2003.

A Appendix

A.1 Overview of experiments

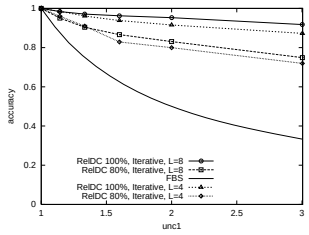

Figure 12: Pub1

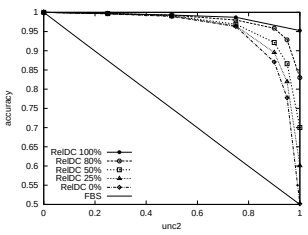

Figure 13: Pub2

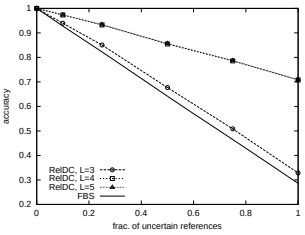

Figure 14: Mov1

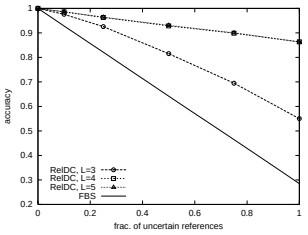

Figure 15: Mov2

The fact that RelDC achieves high cleaning accuracy is captured in Figures 12, 13, 14 and 15 which show the cleaning accuracy of traditional techniques ('FBS') and that of the proposed approach ('RelDC') on publications and movies datasets as various parameters are varied.

A.2 Computing connection strength

The connection strength $c(u, v)$ between nodes u and v should reflect how strongly nodes u and v are related to each other via relationships in the graph G. Many existing measures such as the length of the shortest path or the value of the maximum network flow between nodes u and v

could potentially be used for this purpose. Those measures, however, have certain drawbacks as explained next. Figure 16 shows a small subgraph of a larger graph. The subgraph contains two paths between nodes u and v: $\mathscr{P}_a = u \to a \to v$ and $\mathscr{P}_b = u \to b \to v$. In this figure node b "connects" many

nodes, not just u and v, whereas node a "connects" only u and v. The subgraph in Figure 16 might be, for example, a part of a large publication ARG, similar to that of in Figure 4. That is, nodes u and v may correspond to two authors, node a to a specific publication, and node b to a university which connects numerous authors. Intuitively, we expect that the connection strength between u and v via node a is stronger than the connection strength between u and v via node b: since b connects many nodes it is not surprising we can also connect u and v via b as well, whereas the connection via a is unique to u and v. Therefore, it is natural to require that the connection strengths $c(\mathscr{P}_a)$ and $c(\mathscr{P}_b)$ of paths $\mathscr{P}_a = u \to a \to v$ and $\mathscr{P}_b = u \to b \to v$ be such that $c(\mathscr{P}_a) > c(\mathscr{P}_b)$. Notice, many measures such as *path length*, *network flow* fail to achieve this requirement.

A natural way to compute the connection strength $c(u, v)$ between node u and v (which does not suffer from the above drawback) is to compute it as the probability to reach node v from node u via random walks in graph G. Each step of the random walk is done according to a certain probability. Let us now show how PM model computes $c(\mathscr{P}_a)$ and $c(\mathscr{P}_b)$. To capture the fact that $c(\mathscr{P}_a) > c(\mathscr{P}_b)$, values of $c(\mathscr{P}_a)$ and $c(\mathscr{P}_b)$ are computed as the probabilities to follow paths $\mathscr{P}_a$ and $\mathscr{P}_b$ respectively as follows. For path $\mathscr{P}_b$ we start from u. Next we have a choice to go to a or b with probabilities of $\frac{1}{2}$, and we choose to follow (u, b) edge. From node b we can go to any of the $N-1$ nodes (cannot go back to u) but we go specifically to v. So the probability to reach v via path $\mathscr{P}_b$ is $\frac{1}{2(N-1)}$. For path $\mathscr{P}_a$ we start from u, we go to a with probability $\frac{1}{2}$ at which point we have no choice but to go to v, so the probability to follow $\mathscr{P}_a$ is $\frac{1}{2}$.

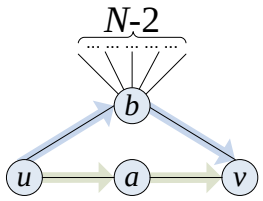

Figure 16: Motivating connection strength formula

A.3 Further improvement of the independent case formulae

A.3.1 Handling weight-1 edges.

The formula in Equation (9) assumes 2^{n_i} iterations will be needed to compute $\mathrm{P}(E_i^{\rightarrow}|\mathscr{P}^{\exists})$. This formula can be modified further to achieve more efficient computation as follows. In practice, some of the p_{ij}'s, or even all of them, are often equal to 1. Figure 17 shows the case where m ($0 \le m \le n_i$) edges incident to node v_i are labeled with 1. Let $\widetilde{\mathbf{a}}_i$ denote vector $(a_{i0}, a_{i1}, \ldots, a_{i(n_i-m)})$ and let $\widetilde{\mathcal{A}}_i$ be the set of all possible instantiations of this vector. Then Equation (9) can be simplified to

$$\mathrm{P}(E_i^{\rightarrow}|\mathscr{P}^{\exists}) = \sum_{\widetilde{\mathbf{a}}_i \in \widetilde{\mathcal{A}}_i} \left\{ \left[\frac{1}{m + \sum_{j=0}^{n_i-m} a_{ij}} \right] \left[\prod_{j=0}^{n_i-m} p_{ij}^{a_{ij}} (1-p_{ij})^{1-a_{ij}} \right] \right\}. \tag{11}$$

The number of iteration is reduced from 2^{n_i} to 2^{n_i-m}.

A.3.2 Computing $\mathbf{P}(E_i^{\rightarrow}|\mathscr{P}^{\exists})$ as $\sum_{l=0}^{n_i} \frac{1}{1+l}\mathbf{P}(s_i = l)$.

Performing 2^{n_i-m} iterations can still be expensive for the cases when $(n_i - m)$ is large. Next we discuss several ways to deal with this issue.

Firstly, while in general the value of 2^{n_i-m} can be large, for a particular pARG it can be that (a)2^{n_i-m} is never large or (b) 2^{n_i-m} can be large but bearable[3] and the cases when it is large are infrequent. In those cases further simplification might not be required.

Secondly, if the above is not applicable, in many situations the answer can be quickly estimated (rather than computed exactly) using results from Poisson trials theory. Let us denote the following sum as s_i: $s_i = \sum_{j=1}^{n_i} a_{ij}$. From a basic probability course we know that the binomial distribution gives the number of successes in n independent trials where each trial is successful with the same probability p [9]. The binomial distribution can be viewed as a sum of several *i.i.d.* Bernoulli trials. The *Poisson trials* process is similar to the binomial distribution process where trials are still independent but not necessarily identically

[3]Item (b) is important when $c(u, v)$ needs to be computed multiple times for different nodes.

distributed, i.e. the probability of success in the i-th trial is p_i. We can modify Equation (10) to compute $\mathrm{P}(E_i^{\rightarrow}|\mathscr{P}^{\exists})$ as follows:

$$\mathrm{P}(E_i^{\rightarrow}|\mathscr{P}^{\exists}) = \sum_{l=0}^{n_i} \frac{1}{1+l}\mathrm{P}(s_i = l). \tag{12}$$

Notice, for a given i we can treat $a_{i1}, a_{i2}, \ldots, a_{in_i}$ as a sequence of n_i Bernoulli trials with probabilities of success $\mathrm{P}(a_{ij} = 1) = p_{ij}$. One would want to *estimate* $\mathrm{P}(s_i = l)$ *quickly*, rather than compute it exactly via iterating over all cases when $(s_i = l)$. That is, we would like to avoid computing $\mathrm{P}(s_i = l)$ as

$$\mathrm{P}(s_i = l) = \sum_{\substack{\mathbf{a}_i \in \mathcal{A}_i \\ s_i = l}} \prod_{j=0}^{n_i} p_{ij}^{a_{ij}} (1 - p_{ij})^{1-a_{ij}}.$$

There are multiple cases when $\mathrm{P}(s_i = l)$ can be computed quickly by various methods. Different methods might be applicable in different situations and produce estimations of different quality. For example, in certain cases it can be possible to utilize the Poisson trials theory to estimate $\mathrm{P}(s_i = l)$. For instance, if each p_{ij} is small then from the probability theory we know that

$$\mathrm{P}(s_i = l) = \frac{\lambda^l e^{-\lambda}}{l!}\left\{1 + O\left(\lambda \max_{j=1,2,\ldots,n_i} p_{ij} + \frac{l^2}{\lambda} \max_{j=1,2,\ldots,n_i} p_{ij}\right)\right\}, \text{ where } \lambda = \sum_{j=1}^{n_i} p_{ij}. \tag{13}$$

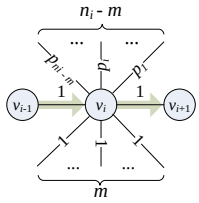

Figure 17: Probability to follow edge $E_i = (v_i, v_{i+1})$

An example of a more coarse estimation is the following (linear cost) approximation formula. First we need to compute the expected number of edges μ_i among n_i edges $E_{i1}, E_{i2}, \ldots, E_{in_i}$, where $\mathrm{P}(E_{ij}^{\exists}) = p_{ij}$, as follows: $\mu_i = m + \sum_{j=1}^{n_i - m} p_{ij}$. Then since there are $1 + \mu_i$ possible links to follow on average, the probability to follow E_i can be coarsely estimated as

$$\mathrm{P}(E_i^{\rightarrow}|\mathscr{P}^{\exists}) \approx \frac{1}{1+\mu_i} = \frac{1}{m + \sum_{j=0}^{n_i - m} p_{ij}}. \tag{14}$$

Numerous other methods exist to achieve this goal.

A.4 Options of the same choice node on the path

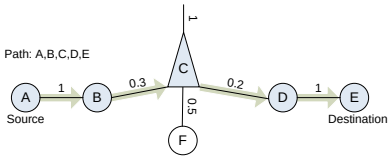

Figure 18: Choice node on the path: illegal path

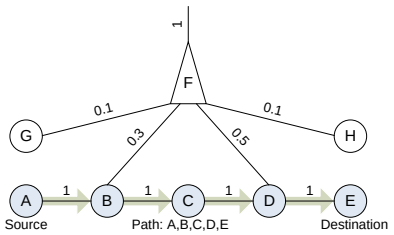

Figure 19: Options of the same choice on the path

Let us define two sets $\mathbf{f}$ and $\mathbf{d}$ – of 'free' and 'dependent' a_{ij}'s as follows:

$$\begin{aligned} \mathbf{f} &= \{a_{ij} : \forall r, s \ (r \neq i \text{ or } s \neq j) \Rightarrow dep(E_{ij}^{\exists}, E_{rs}^{\exists}) = \texttt{false}\}, \\ \mathbf{d} &= \{a_{ij} : \exists r, s \ (r \neq i \text{ or } s \neq j) : dep(E_{ij}^{\exists}, E_{rs}^{\exists}) = \texttt{true}\}. \end{aligned} \tag{15}$$

Notice, $\mathbf{a} = \mathbf{f} \cup \mathbf{d}$ and $\mathbf{f} \cap \mathbf{d} = \emptyset$. If $\mathbf{d} = \emptyset$, then there is no dependence and the solution is given by Equation (10), otherwise we proceed as follows. Similarly to $\mathbf{a}_i$ we can define $\mathbf{f}_i$ and $\mathbf{d}_i$ as

$$\begin{aligned} \mathbf{f}_i &= \{a_{ij} : a_{ij} \in \mathbf{f}, j = 0, 1, \ldots, n_i\}, \\ \mathbf{d}_i &= \{a_{ij} : a_{ij} \in \mathbf{d}, j = 1, 2, \ldots, n_i\}. \end{aligned} \tag{16}$$

Notice, $\mathbf{a}_i = \mathbf{f}_i \cup \mathbf{d}_i$ and $\mathbf{f}_i \cap \mathbf{d}_i = \emptyset$. We define $\mathcal{D}$ as the set of all possible instantiations of $\mathbf{d}$, and $\mathcal{F}_i$ as the set of all possible instantiations of $\mathbf{f}_i$. Then

$$\mathrm{P}(\mathscr{P}^{\rightarrow}) = \underbrace{\left(\prod_{i=1}^{k-1} q_i\right)}_{\mathrm{P}(\mathscr{P}^{\exists})} \sum_{\mathbf{d}\in\mathcal{D}} \Bigg\{ \underbrace{\left[\prod_{i=1}^{k-1}\left(\sum_{\mathbf{f}_i\in\mathcal{F}_i}\left\{\left[\frac{1}{\sum_{j=0}^{n_i} a_{ij}}\right]\left[\prod_{j:a_{ij}\in\mathbf{f}_i} p_{ij}^{a_{ij}}(1-p_{ij})^{1-a_{ij}}\right]\right\}\right)\right]}_{\Psi(\mathbf{d})} \mathrm{P}(\mathbf{d}) \Bigg\}. \tag{17}$$

Equation (17) iterates over all feasible instantiations of $\mathbf{d}$. $\mathrm{P}(\mathbf{d})$ is the probability of a specific instance of $\mathbf{d}$ to occur. Equation (17) contains term $\sum_{\mathbf{d}\in\mathcal{D}}\{\Psi(\mathbf{d})\mathrm{P}(\mathbf{d})\}$. What this achieves is that a particular instantiation of $\mathbf{d}$ "fixates" a particular combination of all "dependent" edges, and $\mathrm{P}(\mathbf{d})$ corresponds to the probability of that combination. Notice, $\Psi(\mathbf{d})$ directly corresponds to $\mathrm{P}(\mathscr{P}^{\rightarrow}|\mathscr{P}^{\exists})$ part of Equation (10). To compute $\mathrm{P}(\mathscr{P}^{\rightarrow})$ in Equation (17), we only need to specify how to compute $\mathrm{P}(\mathbf{d})$.

Computing P(d) Recall, we now consider the cases where a_{ij} is in $\mathbf{d}$ only because there is (at least one) another $a_{rs} \in \mathbf{d}$ such that $dep(E_{ij}^{\exists}, E_{rs}^{\exists}) = \texttt{true}$ and $v_{E_{ij}}^* = v_{E_{rs}}^*$. Figure 7 is an example of such a case. So, for each $a_{ij} \in \mathbf{d}$ we can identify choice node $v_l^* = v_{E_{ij}}^*$ and compute set $C_l = \{a_{rs} \in \mathbf{d} : v_{E_{rs}}^* = v_l^*\}$. Then, for any two distinct elements $a_{ij} \in C_l$ and a_{rs} the following holds: $dep(E_{ij}^{\exists}, E_{rs}^{\exists}) = \texttt{true}$ if and only if $a_{rs} \in C_l$.

In other words, we can split set $\mathbf{d}$ into non intersecting subsets $\mathbf{d} = C_1 \cup C_2 \cup \cdots \cup C_m$. The existence of each edge E_{ij} such that a_{ij} is in one of those sets C_l depends only on the existence of those edges E_{rs}'s whose a_{rs} is in C_l as well. Therefore $\mathrm{P}(\mathbf{d})$ can be computed as $\mathrm{P}(\mathbf{d}) = \mathrm{P}(\mathbf{d}_{C_1})\mathrm{P}(\mathbf{d}_{C_2}) \times \cdots \times \mathrm{P}(\mathbf{d}_{C_m})$, where $\mathbf{d}_{C_l}$ is a particular instantiation of a_{ij}'s from C_l. Now, to be able to compute Equation (17), we only need to specify how to compute $P(\mathbf{d}_{C_l})$ $(l = 1, 2, \ldots, m)$.

Computing $P(\mathbf{d}_{C_l})$. Figure 20 shows choice node v_l^* with n options $u_1, u_2, \ldots, u_n$. Each (v_l^*, u_j) edge $(j = 1, 2, \ldots, n)$ is labeled with probability p_j. As before, to specify which edge is present and which is

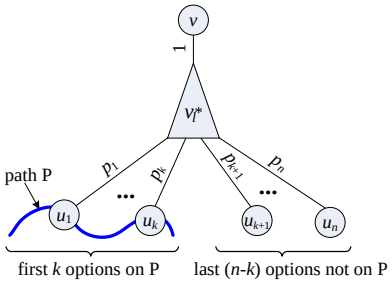

Figure 20: Intra choice dependence.

absent, each option edge has variable a_j associated with it. Variable $a_j = 1$ if and only if the edge labeled with p_j is present, otherwise $a_j = 0$. That is, $\mathrm{P}(a_j = 1) = p_j$ and $p_1 + p_2 + \cdots + p_n = 1$.

Let us assume, without loss of generality, that the first k $(2 \le k \le n)$ options $u_1, u_2, \ldots, u_k$ of v_l^* belong to path $\mathscr{P}$ while the other $(n-k)$ options $u_{k+1}, u_{k+2}, \ldots, u_n$ do not belong to $\mathscr{P}$, as shown in Figure 20. In the context of Figure 20, computing $P(\mathbf{d}_{C_l})$ is equivalent to computing the probability a particular instantiation of vector $(a_1, a_2, \ldots, a_k)$ to occur.

Notice, only one a_i among $a_1, a_2, \ldots, a_k, a_{k+1}, a_{k+2}, \ldots, a_n$ can be 1, the rest are zeroes. First let us compute the probability of instantiation $a_1 = a_2 = \cdots = a_k = 0$. For that case one of $a_{k+1}, a_{k+2}, \ldots, a_n$ should be equal to 1. Thus $\mathrm{P}(a_1 = a_2 = \cdots = a_k = 0) = p_{k+1} + p_{k+2} + \cdots + p_n$.

The second case is when one of $a_1, a_2, \ldots, a_k$ is 1. Assume $a_j = 1$ $(1 \le j \le k)$, then $\mathrm{P}(a_j = 1) = p_j$. To summarize:

$$\mathrm{P}(a_1, a_2, \ldots, a_k) = \begin{cases} p_j & \text{if } \exists j\ (1 \le j \le k) : a_j = 1; \\ p_{k+1} + p_{k+2} + \cdots + p_n & \text{otherwise.} \end{cases}$$

Now we know how to compute $P(\mathbf{d}_{C_l})$ $(l = 1, 2, \ldots, m)$, thus we can compute $\mathrm{P}(\mathbf{d})$. Therefore we have specified how to compute path connection strength using Equation (17).

A.5 Computing the total connection strength

Let us give the motivation of why the *summation* of individual simple paths is performed. We associate the connection strength between two nodes u and v with probability of reaching v from u via only L-short simple paths. Let us name those simple paths $\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_k$, Let us call $\mathcal{G}(u, v)$ the subgraph comprised of the union of those paths: $\mathcal{G}(u, v) = \mathscr{P}_1 \cup \mathscr{P}_2 \cup \cdots \cup \mathscr{P}_k$. Subgraph $\mathcal{G}(u, v)$ is a subgraph of the complete graph $G = (V, E)$, where V is the set of vertices $V = \{v_i : i = 1, 2, \ldots, |V|\}$ and E is the set of edges $E = \{E_i : i = 1, 2, \ldots, |E|\}$. Let us define a_i as follows: $a_i = 1$ if and only if edge E_i is present, otherwise $a_i = 0$. Let $\mathbf{a}$ denote vector $(a_1, a_2, \ldots, a_{|E|})$ and let $\mathcal{A}$ be the set of all possible instantiations of $\mathbf{a}$.

We need to compute the probability to reach v from u via subgraph $\mathrm{P}(\mathcal{G}(u, v)^{\rightarrow})$ which we treat as the measure of the connection strength. We can represent $\mathrm{P}(\mathcal{G}(u, v)^{\rightarrow})$ as

$$\mathrm{P}(\mathcal{G}(u, v)^{\rightarrow}) = \sum_{\mathbf{a} \in \mathcal{A}} \mathrm{P}(\mathcal{G}(u, v)^{\rightarrow} | \mathbf{a}) \mathrm{P}(\mathbf{a}). \tag{18}$$

Notice, when computing $\mathrm{P}(\mathcal{G}(u, v)^{\rightarrow} | \mathbf{a})$ we assume a particular instantiation of $\mathbf{a}$. So the complete knowledge of which edges are present and which are absent is available, as if all the edges were "fixed". Assuming one particular instantiation of $\mathbf{a}$, there is no dependence among edge existence events any longer: each edge is either present with 100% probability or absent with 100% probability. Thus

$$\mathrm{P}(\mathcal{G}(u, v)^{\rightarrow} | \mathbf{a}) = \sum_{i=1}^{k} \mathrm{P}(\mathscr{P}_i^{\rightarrow} | \mathbf{a}), \tag{19}$$

and

$$\begin{aligned} \mathrm{P}(\mathcal{G}(u, v)^{\rightarrow}) &= \sum_{\mathbf{a} \in \mathcal{A}} \mathrm{P}(\mathcal{G}(u, v)^{\rightarrow} | \mathbf{a}) \mathrm{P}(\mathbf{a}) \\ &= \sum_{\mathbf{a} \in \mathcal{A}} \left[\left(\sum_{i=1}^{k} \mathrm{P}(\mathscr{P}_i^{\rightarrow} | \mathbf{a}) \right) \mathrm{P}(\mathbf{a}) \right] \\ &= \sum_{i=1}^{k} \left[\sum_{\mathbf{a} \in \mathcal{A}} \Big(\mathrm{P}(\mathscr{P}_i^{\rightarrow} | \mathbf{a}) \mathrm{P}(\mathbf{a}) \Big) \right] \\ &= \sum_{i=1}^{k} \mathrm{P}(\mathscr{P}_i^{\rightarrow}). \end{aligned} \tag{20}$$

Equation (20) shows that the total connection strength is the sum of the connection strength of all L-short simple paths.