A

Improving Classification Quality in Uncertain Graphs

Michele Dallachiesa, University of Trento
Charu C. Aggarwal, IBM T.J. Watson Research Center
Themis Palpanas, Paris Descartes University

In many real applications that use and analyze networked data, the links in the network graph may be erroneous, or derived from probabilistic techniques. In such cases, the node classification problem can be challenging, since the unreliability of the links may affect the final results of the classification process. If the information about link reliability is not used explicitly, the classification accuracy in the underlying network may be affected adversely. In this paper, we focus on situations that require the analysis of the uncertainty that is present in the graph structure. We study the novel problem of node classification in uncertain graphs, by treating uncertainty as a first-class citizen. We propose two techniques based on a Bayes model and automatic parameter selection, and show that the incorporation of uncertainty in the classification process as a first-class citizen is beneficial. We experimentally evaluate the proposed approach using different real data sets, and study the behavior of the algorithms under different conditions. The results demonstrate the effectiveness and efficiency of our approach.

General Terms: Graphs, Uncertainty, Classification

Additional Key Words and Phrases: Network Classification, Structural Classification, Label Propagation

1. INTRODUCTION

The problem of collective classification is a widely studied one in the context of graph mining and social networking applications. In this problem, we have a network containing nodes and edges. A subset of the nodes in this network may be labeled. Typically, such labels may represent some properties of interest in the underlying network. Some examples of such labeled networks in real scenarios are listed below:

— In a bibliographic network, nodes correspond to authors, and the edges between them correspond to co-authorship links. The labels in the bibliographic network may correspond to subject areas that experts are interested in. It is desirable to use this information in order to classify other nodes in the network.
— In a biological network, the nodes correspond to the proteins. The edges may represent the possibility that the proteins may interact. The labels may correspond to properties of proteins [3].
— In a movie-actor network, the nodes correspond to the actors. The edges correspond to the co-actor relationship between the different actors. The labels correspond to the pre-dominant genre of the movie of the actor.
— In a patent network, the nodes correspond to patent assignees. The edges model the citations between the respective patents. The labels correspond to the class categories.

This work is an extended version of the SSDBM paper published in [26].
Author's addresses: M. Dallachiesa, University of Trento; C. Aggarwal, IBM T.J. Watson Research Center; T. Palpanas, Paris Descartes University.

DOI: http://dx.doi.org/10.1145/0000000.0000000

In such networks, only a small fraction of the nodes may be labeled, and these labels may be used in order to determine the labels of other nodes in the network. This problem is popularly referred to as *collective classification* or *label propagation* [19; 21; 22; 38; 30; 47; 48; 49], and a wide variety of methods have been proposed for this problem.

In recent years, an increasing amount of data is uncertain or of low quality because of how it is extracted. In such cases, the low quality of the data has a direct impact on the data mining results. In many scenarios, the data is extracted with the use of probabilistic modeling processes [8; 36; 42]; as a result one can model the data attributes in the form of probability distributions. Modeling the data in the form of probability distributions has the advantage that uncertainty can be used as a first class citizen in the mining process. Such an approach has been widely studied in the database literature [20], and is known to have several advantages.

Recently, this approach has also found increasing interest and acceptance in the context of network data [52; 53; 4; 17; 5]. In many real networks, the links[1] are uncertain in nature, and are derived with the use of a probabilistic process. In such cases, a probability value may be associated with each edge. Some examples are as follows:

— In biological networks, the links are derived from probabilistic processes. In such cases, the edges have uncertainty associated with them. Nevertheless, such probabilistic networks are valuable, since the probability information on the links provides important information for the mining process.
— The links in many military networks are constantly changing and may be uncertain in nature. In such cases, the analysis needs to be performed with imperfect knowledge about the network.
— Networks in which some links have large failure probabilities are uncertain in nature.
— Many human interaction networks can be created from real interaction processes, and such links are often uncertain in networks.

Thus, such networks can be represented as probabilistic networks, in which we have probabilities associated with the existence of links. Such probabilities can be very useful for improving the effectiveness of problems such as collective classification. Furthermore, these networks may also have properties associated with nodes, that are denoted by labels. Recent years have seen the emergence of numerous methods for uncertain graph management [29; 32; 44; 15; 10] and mining [32; 34; 31; 37; 43; 50; 51], in which uncertainty is used directly as a first-class citizen. However, none of these methods address the problem of collective graph classification.

One possibility is to use *sampling* of possible worlds on the edges in order to generate different instantiations of the underlying network. The collective classification problem can be solved on these different instantiations, and voting can be used in order to report the final class label. The major disadvantage with this approach is that the sampling process could result in a sparse or disconnected network which is not suited to the collective classification problem. In such cases, good class labels cannot be easily produced with a modest number of samples.

In this paper, we investigate the problem of collective classification in uncertain networks with a more direct use of the uncertainty information in the network. We design two algorithms for collective classification. The first algorithm uses a probabilistic approach, which explicitly accounts for the uncertainty in the links in the classification. The second algorithm works with the assumption that most of the information in the network is encoded in high-probability links, and low-probability links sometimes even degrade the quality. Therefore, the algorithm uses the links with high probability in

[1] In the rest of this paper we use the terms *network* and *graph*, as well as *link* and *edge*, interchangeably.

earlier iterations, and successively relaxes the constraints on the quality of the underlying links. The idea is that a greater caution in early phases of the algorithm ensures convergence to a better optimum.

In summary, we make the following contributions.

— We introduce the problem of collective classification in uncertain graphs, where uncertainty is associated with the edges of the graph, and provide a formal definition for this problem.
— We introduce two algorithms based on iterative probabilistic labeling that incorporate the uncertainty of edges in their operation. These algorithms are based on a Bayes formulation, which enables them to capture correlations across different classes, leading to improved accuracy.
— We perform an extensive experimental evaluation, using two real datasets from diverse domains. We evaluate our techniques using a multitude of different conditions, and input data characteristics. The results demonstrate the effectiveness of the proposed techniques and serve as guidelines for the practitioners in the field.

This paper is organized as follows. In Section 2 we survey prior studies on collective classification and on mining uncertain networks. In Section 3, we formally define the problem of collective classification in uncertain networks. In Section 4, we present our model and two algorithms for collective classification. We discuss the space and time complexity of our proposal in Section 5, and we present the results of our experimental evaluation in Section 6. Finally, we discuss the conclusions in Section 7.

2. RELATED WORK

The problem of node classification has been studied in the graph mining literature, and especially relational data in the context of *label or belief propagation* [45; 47; 48]. Such propagation techniques are also used as a tool for semi-supervised learning with both labeled and unlabeled examples [49]. Collective classification [39; 38; 21] refers to semi-supervised learning methods that exploit the network structure and node class labels to improve the classification accuracy. These techniques are mostly based on the assumption of homophily in social networks [24; 41]: neighboring nodes tend to belong to the same class. A technique has been proposed in [38], which uses link-based similarity for node-classification in directed graphs.

Collective classification methods have also been used in the context of blogs [21]. In this study, the authors propose two algorithms: the first is based on the assumption that ”nodes link to other nodes with similar labels”. Confidence values for all classes on the labeled nodes, letting labels to change over time. Labels assigned during the first iterations are expected to be more accurate, and have a greater influence on the final assignment. The second is based on an adaptation of the classical k_{th} nearest neighbor classifier and it is based on the principle ”nodes with similar neighborhoods have similar labels”. The results show that there is nearly no difference between the two algorithms in terms of accuracy.

In [22], Bilgic et al. discuss the problem of overcoming the propagation of erroneous labels by asking the user for more labels. A method for performing collective classification of email speech acts has been proposed by Carvalho et al. in [25], exploiting the sequential correlation of emails. In [30], Ji et al. integrate the classification of nodes in heterogeneous networks with ranking. Methods for leveraging label consistency for collective classification have been proposed in [47; 48; 49].

Recently, the database and data mining community has investigated the problem of uncertain data mining widely [20; 16]. A comprehensive review of the proposed models and algorithms can be found in [7]. Several database systems supporting uncertain data have been proposed, such as Conquer [9], Trio [11], MistiQ [6], MayMBS [12]

and Orion [13]. The problem of integrating uncertain data from multiple sources is discussed in [40]. Uncertain data can be explicitly modeled in a probabilistic way in a wide variety of ways, such as the use of mathematical models, as a byproduct of the cleansing process [42], data acquisition [8], or during the estimation of missing values [36]. For example, when a Bayesian model is used to estimate missing values, the resulting values can be naturally expressed in the form of a probability distribution [36].

The ”possible worlds” model, introduced by Abiteboul et al. [14], formalizes uncertainty by defining the space of the possible instantiations of the database. Instantiations must be consistent with the semantics of the data. For example, in a graph database representing moving object trajectories there may be be different configurations of the edges where each node represents a region in the space. However, an edge cannot connect a pair of nodes that represent a pair of non-neighboring regions. The main advantage of the ”possible worlds” model is that the formulations of the queries originally designed to cope with certain data can be directly applied on each possible instantiation. Many different alternatives have then been propose to aggregate the results across the different instantiations.

Despite its attractiveness, the number of possible worlds explodes very quickly and even their enumeration becomes intractable problem. To overcome these issues, simplifying assumptions have been introduced to leverage its simplicity: The tuple- and the attribute-uncertainty models [18; 7]. In the attribute-uncertainty model, the uncertain tuple is represented by means of multiple samples drawn from its Probability Density Function (PDF). In contrast, in the tuple-uncertainty model the value of the tuple is fixed but the tuple itself may not exist.

Similar simplifications have been considered for graph databases where nodes may or may not exist (node-uncertainty) and edges are associated with an existence probability (edge-uncertainty). The underlying uncertainty model can then be used to generate graph instances, eventually considering additional generation rules to consider correlations across different nodes and edges. In this study we combine a Bayes approach and the edge-uncertainty model.

It has been widely recognized that the presence of noise and uncertainty significantly impacts the quality of the results, unless the quality of the data is explicitly accounted for in the mining process [20; 23; 27]. In the context of multidimensional data, uncertain data management and mining methods are surveyed in [20]. The problem of uncertain graph mining has also been investigated extensively. The most common problems studied in uncertain graph management are those of nearest neighbor query processing [29; 44], reachability computation [33] and subgraph search [46]. In the context of uncertain graph *mining*, the problems commonly studied are frequent subgraph mining [43; 50; 51], reliable subgraph mining [32], and clustering [31; 37]. Recently, the problem of graph classification has also been studied for the uncertain scenario [34], though these methods are designed for classification of many small graphs, in which labels are attached to the entire graph rather than a node in the graph. Typical social and web-based scenarios use a different model of collective classification, in which the labels are attached to nodes in a single large graph.

In this work, we study the problem of *collective classification* in the context of uncertain networks, where the underlying links are uncertain. Uncertainty impacts negatively on the classification accuracy. First, links may connect sub-networks of very different density, causing the propagation of erroneous labels. Second, the farthest distance between two nodes tends to be smaller in very noisy networks, because of the presence of a larger number of uncertain edges, which include both true and spurious edges. This reduces the effectiveness of iterative models because of the faster propaga-

tion of errors. Some of our techniques, which drop uncertain links at earlier stages of the algorithm, are designed to ameliorate these effects.

3. COLLECTIVE CLASSIFICATION PROBLEM

In this section, we formalize the problem of collective classification after introducing some definitions. An uncertain network is composed of nodes whose connections may exist with some probability.

Definition 3.1 (*Uncertain Network*). An uncertain network is denoted by $G = (N, A, P)$, with node set N, edge set A and probability set P. Each edge $(i, j) \in A$ is associated with a probability value $p_{ij} \in P$. This is the probability that edge (i, j) exists in the network.

We assume that the network is undirected, though the method can easily be extended to the directed scenario. We can assume that the $|N| \times |N|$ matrix P has entries which are denoted by p_{ij} and $p_{ij} = p_{ji}$. A node $i \in N$ can be associated with a label, representing its membership in a class. For ease in notation, we assume that node labels are integers.

Definition 3.2 (*Node Label*). Given a set of labels S drawn from a set of integers $\{1 \ldots l\}$, we denote the label of node i by $L(i)$. If a node i is unlabeled, the special label 0 is used.

We can now introduce the definition of the collective classification problem on uncertain graphs.

PROBLEM 3.1 (UNCERTAIN COLLECTIVE CLASSIFICATION). *Given an uncertain network $G = (N, A, P)$ and the subset of labeled nodes $T_0 = \{i \in N : L(i) \neq 0\}$, predict the labels of nodes in $N - T_0$.*

Figure 1 shows an example of an uncertain network. Nodes 1, 2, and 3 are labeled *white*, and nodes 5, 7, and 8 are labeled *black*. The label of nodes 4 and 6 is unknown. The aim of collective classification is to assign labels to nodes 4 and 6.

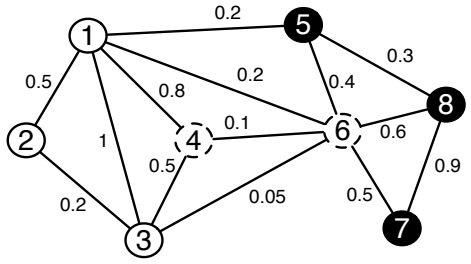


Fig. 1. **Example of uncertain network. Nodes $\{1, 2, 3\}$ are labeled *white* and nodes $\{5, 8, 7\}$ are labeled *black*, while labels for nodes $\{4, 6\}$ are unknown. Edges between nodes exist with some probability.**

4. ITERATIVE PROBABILISTIC LABELING

In this section, we first present the algorithm for iterative probabilistic labeling. A Bayes approach is used in order to perform the iterative probabilistic labeling. This method models the probabilities of the nodes belonging to different classes on the basis of the adjacency behavior of the nodes. The Bayes approach can directly incorporate the

edge uncertainty probabilities into the estimation process. We continue with a second algorithm that builds upon the first one, and is based on iterative edge augmentation. Finally, we describe a variation of the second algorithm that is a linear combination of two classifiers.

4.1. Bayes Approach

The overall approach for the labeling process uses a Bayesian model for the labeling and, similarly to [39], is inspired by label propagation. In the rest of the paper, we refer to this algorithm as *uBayes*. Given that we have an unlabeled node r, which is adjacent to s other nodes denoted by $t_1 \dots t_s$, how do we determine the label of the node r? It should be noted that the concept of adjacency is also uncertain, because the edges are associated with probabilities of existence. This is particularly true, when the edge probabilities are relatively small, since the individual network instantiations are likely to be much sparser and different than the probabilistic descriptions. Furthermore, for each edge (i, j) we need to estimate the probability of the node j having a particular label value, given the current value of the label at node i. This is done with the use of training data containing the labels and edges in the network. These labels and edges can be used to construct a Bayesian model of how the labels on the nodes and edges relate to one another.

The algorithm uses an iterative approach, which successively labels more nodes in different iterations. This is the set T of nodes whose labels will not be changed any further by the algorithm. Initially, the algorithm starts off by setting T to the initial set of (already) labeled nodes T_0. The set in T is expanded to $T \cup T^+$ in each iteration, where T^+ is the set of nodes not yet labeled that are adjacent to the labeled nodes in T. If T^+ is empty, either all nodes have been labeled or there is a disconnected component of the network whose nodes are not in T_0.

The expanded set of labeled nodes are added to the set of training nodes in order to compute the propagation probabilities on other edges. Thus, the overall algorithm iteratively performs the following steps:

— Estimating the Bayesian probabilities of propagation from the current set of edges.
— Computing the probabilities of the labels of the nodes in $N - T$.
— Expanding the set of the nodes in T, by adding the set of nodes from T^+, whose labels have the highest probability for a particular class.

These steps are repeated until no more nodes reachable from the set T remain to be labeled. We then label all the remaining nodes in a single step, and terminate. The overall procedure for performing the analysis is illustrated in Algorithm 1. It now remains to discuss how the individual steps in Algorithm 1 are performed.

The two most important steps are the computation of the edge-propagation probabilities and the expansion of the node labels with the use of the Bayes approach. For a given edge (i, j) we estimate $P(L(i) = p|L(j) = q)$. This is estimated from the data in *each iteration* by examining the labels of nodes that have already been decided. Therefore, the training process is successively refined in each iteration. The value of $P(L(i) = p|L(j) = q)$ can be estimated by examining those edges, for which one end point contains a label of q. Among these edges, we compute the fraction for which the other end point contains a label of p. For example, in the network shown in Figure 1, the probability $P(L(6) = black|L(5) = black)$ is estimated as $(0.3 + 0.9)/(0.3 + 0.9 + 0.2) = 0.85$. The label of node 6 is unknown, and it is not considered in the calculation. Note that this is simply equal to the probability that both end points of an edge are *black*, if one of them is *black*. Therefore, one can compute the

Algorithm *uBayes*(Graph: G, Uncertainty Prob.: P, Initial Labeling: T_0);

begin
  $T = T_0$;
  while (not termination) **do**
  begin
    For all edges (i, j), $j \notin T$, $j \in T$
      Compute edge propagation probabilities, $P(L(i) = p|L(j) = q)$;
    For all nodes $r \in \{N - T\}$
      Compute label for node r using Bayes rule:
      $L(r) = argmax_{p \in S} P(L(r) = p) \cdot \prod_k P(L(i_k) = t_k|L(r) = p)$;
    Expand T with T^+, the set of all unlabeled nodes adjacent to labeled nodes in T;
  end
end

Algorithm 1: Broad Framework for Uncertain Classification.

uncertainty weighted conditional probabilities for this in the training process of each iteration.

This provides an estimate for the conditional probability. We note that in some cases, the number of nodes with a label of either p or q may be too small for a robust estimation. The following smoothing techniques are useful in reducing the effect of ill-conditioned probabilities:

— We always add a small value δ to each probability. This is similar to Laplace smoothing and prevents any probability value from being zero, which would cause problems in a multiplicative Bayes model (by turning the entire probability to zero).
— In some cases, the estimation may not be possible when labels do not exist for either nodes p or q. In those cases, we set the probabilities to their prior values.

The prior is defined as the value of $P(L(i) = p)$, and is equal to the fraction of currently labeled nodes with label of p. The prior therefore defines the default behavior in cases where the adjacency information cannot be reasonably used in order to obtain a better posteriori estimation.

For an unlabeled node r, whose neighbors $i_1 \ldots i_s$ have labels $t_1 \ldots t_s$, we estimate its (unnormalized) probability by using the naive Bayes rule over all the adjacent labeled neighbors. This is therefore computed as follows:

$$P(L(r) = p|L(i_1) = t_1 \ldots L(i_s) = t_s) \propto P(L(r) = p) \cdot \prod_k P(L(i_k) = t_k|L(r) = p)$$

Note that the above model incorporates the uncertainty probabilities directly within the product term of the equation. We can perform the estimation for each of the different classes separately. If desired, one can normalize the probability values to sum to one. However, such a normalization is not necessary in our case, since the only purpose of the computation is to determine the highest probability value in order to assign labels.

4.2. Iterative Edge Augmentation

The approach mentioned above is not very effective when a large fraction of the edges are noisy. In particular, if many edges have a low probability, this can have a significant impact on the classification process.

Figure 2 shows an example. Nodes 1, 2, are labeled *white*, and nodes 3, 4, 6, 7, 8 and 9 are labeled *black*. The label of node 5 is unknown and must be assigned by the algorithm. We observe that ignoring the edges whose existence probability is lower than 0.5 is beneficial for the correct classification of node 5.

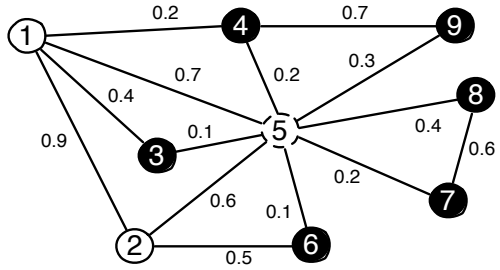


Fig. 2. **Example of uncertain network. Nodes 1, 2, are labeled *white*, and nodes 3, 4, 6, 7, 8 and 9 are labeled *black*, while the label of node 5 is unknown and must be assigned by the algorithm. Edges between nodes exist with some probability.**

Therefore, we use an iterative augmentation process in order to reduce the impact of such edges, by instead favoring the positive impact of high quality edges in the collective classification process. The idea is to *activate* only a subset of the edges for use on the modeling process. In other words, edges which are not activated are not used in the modeling. We call this algorithm *uBayes+*.

We adopt a model inspired by automatic parameter selection in machine learning. Note that, analogous to parameter selection, the choice of a particular subset of high quality links, corresponds to a configuration of the network, and we would like to determine an optimal configuration for our approach. In order to do this, we split the set of labeled nodes T_0 into two subsets: a *training set* denoted by T_{train} and a *hold out* set denoted by T_{hold}. The ratio of the T_0 nodes that are assigned to the training set T_{train} is denoted by β, a user-defined parameter.

The purpose of the hold out set is to aid optimal configuration selection by checking the precise value of the parameters at which the training model provides optimal accuracy over the set of nodes in T_{hold}. We use labels of nodes in T_{train} for the learning process, while using labels of nodes in T_{hold} as for the evaluation of accuracy at a particular configuration of the network. (Note that a label is never used for both the training and the hold out set, in order to avoid overfitting.) The idea is to pick the ratio of active edges in such a way so as to optimize the accuracy on the hold out set. This ensures that an optimal fraction of the high quality edges are used for the labeling process.

We start off considering a small fraction of the high probability edges, iteratively expanding the subset of active edges by enabling some of the inactive edges with the highest probabilities. The ratio of active edges is denoted by the parameter θ. Ideally, we want to activate only the edges that contribute positively to the classification of unlabeled nodes. Given a configuration of active edges, we measure their goodness

as the estimated accuracy on labels of nodes in T_{hold}. The value of θ that leads to the highest accuracy, denoted by θ^*, is used as the ratio of edges with the highest probability to activate on the uncertain network G. The resulting network is then used as input for the iterative probabilistic labeling algorithm (*uBayes*).

Despite optimizing accuracy by selecting the best ratio of edges to be considered, the basic model described above is not very efficient, because it requires multiple evaluations of the iterative probabilistic labeling algorithm. In particular, it requires us to vary the parameter θ and evaluate accuracy, in order to determine θ^*.

A more efficient technique for identifying θ^* can be obtained by evaluating the accuracy for different values of θ on a sample of the uncertain network G (rather than the full network). In this case, the sampled graph is obtained by retaining a random set of nodes and the complete set of their edges. This sampling method results in a higher sparsity of the network, that is, the number of distinct paths between any two points of the sampled network can be smaller compared to the original network. Nevertheless, the sampled network retains the topology of the original network: the number of distinct paths between any two random nodes is reduced uniformly [2].

The proposed technique works as follows. We generate a new uncertain network $G' = (N', A', P')$ by sampling $\alpha \cdot |N|$ nodes from G uniformly at random, and retaining the edges from A and probabilities from P referring to these sampled nodes. α is a user-defined parameter that controls the ratio of nodes sampled from G and it implies the size of the sampled uncertain network G'. The initial set of labeled nodes in the sampled uncertain network G' is $T'_0 = T_0 \cap N'$. We split the set of nodes in T'_0 into two random subsets, T'_{train} and T'_{hold}, respectively. The number of nodes in T'_{train} is $\beta \cdot |T'_0|$. We start off considering $\theta|A'|$ edges with the highest probabilities, expanding iteratively the subset of active edges at each iteration by increasing θ. The goodness of parameter θ is estimated as the accuracy of node labels in T'_{hold}. Let θ^* be the value of θ leading to the highest accuracy. We activate $\theta^*|N|$ edges with highest probability in G. The resulting network is then used as input for the iterative probabilistic labeling (Algorithm 1). The overall algorithm is illustrated in Algorithm 2.

We note that the frequencies used to estimate conditional and prior probabilities across the different configurations in Algorithm 2 can be efficiently maintained in an incremental fashion.

4.3. Combining different classifiers

In this section we propose a third algorithm, *uBayes+RN*. It uses an ensemble methodology in order to further improve robustness in scenarios, where some deterministic classifiers can provide good results over *some* subsets of nodes, but not over all the nodes. *uBayes+RN* is the linear combination of two classifiers: the *uBayes+* algorithm and the Relational Neighbor (RN) classifier [39]. The RN classifier is defined as follows:

$$P_{RN}(L(r) = p) = \frac{1}{Z} \sum_{k:L(i_k)=p} p_{i_k r} \tag{1}$$

where $p_{i_k r}$ is the probability and $Z = \sum_k p_{i_k r}$. The *uBayes+* and *RN* algorithms are combined as follows (assuming independence between adjacent labeled neighbors):

$$P(L(r) = p) = \\ P(L(r) = p | L(i_1) = t_1 \dots L(i_s) = t_s) \cdot \delta P_{RN}(L(r) = p)$$

where δ controls the influence of the RN classifier during the collective classification process. When $\delta = 0$ then *uBayes+RN* degenerates to *uBayes+*, while when $\delta = 1$ the

Algorithm *uBayes+*(Graph: G
Uncertainty Prob.: P, Initial Labeling: T_0,
Sampled nodes ratio: α, Train nodes ratio: β);

begin
N' = Random sample of $\alpha \cdot |N|$ nodes from N;
A' = Edges (i, j) in A with $i, j \in N'$;
$(T'_{hold}, T'_{train}) = split(T_0 \cap N', \beta)$;
$F = \theta \cdot |A'|$ edges in A' with
greatest existence probability;
while $(F \neq A')$ **do**
begin
Construct graph $G^F = (N', F)$;
$uBayes(G^F, P, T'_{train})$;
Test accuracy using nodes in T'_{hold};
Expand edges in F with top edges in A';
end
Construct graph $G^* = (N, F)$ with best
configuration (corresponding to θ^*);
$uBayes(G^*, P, T_0)$;
end

Algorithm 2: Iterative Edge Augmentation for Uncertain Classification

two classifiers are weighted equally. Note that this is a simple linear combination. We used this combination, since it sometimes provides greater robustness in the classification process.

5. COMPLEXITY ANALYSIS

In this section, we discuss the complexity of the proposed algorithms.

We start with *uBayes*, which for the computation of the initial statistics requires $O(|N| + |A|)$ (label priors and conditional label probabilities). We observe that the algorithm requires $O(deg(i)^2)$ time to process each node i in the graph. Assuming that all unlabeled nodes will be labeled in K iterations, the algorithm has an overall time complexity of $O(K \sum_{i \in \{N - T_0\}} deg(i)^2)$, where $deg(i)$ is the degree of node i. Using the handshaking lemma, we can rewrite the above expression as $O(Kmaxdeg|A|)$, where $maxdeg = max\{deg(i)|i \in N\}$.

For algorithm *uBayes+*, the computation of the uncertain network sample $G' = (N', A', P')$ requires $O(|N| + |A|)$. Active edges are maintained using a priority list, whose initialization requires $O(|A|)$. Each iteration of the iterative automatic parameter selection procedure can be decomposed as follows. Algorithm 1 (used by *uBayes+*) requires $O(|N'||A'|)$. Testing the classification accuracy requires $O(|N'|)$. Expanding the set of active edges requires $O(|A'|log(|A'|))$. Summing up, each iteration requires:

$$O(log(|A'|)|N'||A'|). \tag{2}$$

Finally, the last call to Algorithm 1 requires $O(K|N||A|)$. Assuming that the parameter selection procedure terminates after K' iterations, the algorithm cost is $O(|N| + |A| + K'(log(|A'|)|N'||A'|))$. Simplifying, the cost is $O(log(|A|)|N||A|)$. The space complexity is $O(|N||A|)$.

Note that algorithms *uBayes+RN* and *uBayes+* have the same space and time complexity.

6. EXPERIMENTAL RESULTS

In this section, we evaluate the proposed techniques under different settings, in terms of both accuracy and performance.

We implemented all techniques in C++ using the Standard Template Library (STL) and Boost libraries, and ran the experiments on a Linux machine equipped with an Intel Xeon 2.40GHz processor and 16GB of RAM. In order to facilitate other researchers to experiment with our approach, we have publicly released the implementation of our algorithms [1].

The reported times do not include the initial loading time, which was constant over all methods. The results were obtained from 5 independent runs. For all experiments we report the averages and 95% confidence intervals.

6.1. Data Sets

In our experiments, we used two data sets for which edge probabilities can be estimated, as described below.

DBLP: The *DBLP data set* [35] is the most comprehensive citation network of curated records of scientific publications in computer science. In our experiments, we consider the subset of publications from 1980 to 2010. The data set consists of $922,673$ nodes and $3,389,272$ edges. Nodes represent authors and edges represent co-authorship relations. The edge probability is an estimate of the probability that two authors coauthored a paper in a year selected randomly during their period of activity. For example, if a pair of authors published papers in ten different years and they both published papers for twenty years, then their edge probability is 0.5. (We consider the union of their periods of activity.) We used 14 class labels, that represent different research fields in computer science. The corresponding labels and their frequencies are illustrated in Table I. The labels were generated by using a set of top conferences and journals in these areas, and the most frequent label in the author's publications is used as the author's label. In our data set, 16% of the nodes are labeled. The rest were not labeled, because the corresponding authors did not have publications in the relevant conferences and journals.

Table I. Node labels and label priors of the *DBLP* dataset.

Id	Name	Prior probability
C_1	Verification & Testing	0.06758
C_2	Computer Graphics	0.01974
C_3	Computer Vision	0.04144
C_4	Networking	0.1301
C_5	Data Mining	0.09498
C_6	Operating systems	0.06058
C_7	Computer Human Interaction	0.05361
C_8	Software Engineering	0.01935
C_9	Machine Learning	0.1543
C_{10}	Bioinformatics	0.1936
C_{11}	Computing Theory	0.04008
C_{12}	Information Security	0.05364
C_{13}	Information Retrieval	0.044
C_{14}	Computational Linguistics	0.02711

US Patent Data Set: The US Patent data set [28] is a citation network of US utility patents. In our experiments, we consider patents issued from 1970 to 1990. The network contained $108,658$ nodes and $1,059,822$ edges. A node represents a patent assignee

and there is an edge between two assignees if there is at least a patent from one assignee citing a patent from the other assignee. The edge probability is an estimate of the probability that one of the two assignee cites the other assignee. For example, assignee A cites 20 patents of which 5 are assigned to assignee B, then their edge probability is 0.25. A category is assigned to each patent. The most frequent category in the assignee's patents is used as assignee label. Table II reports the label class names and their frequencies. These labels cover 66% of nodes. We used class label as ground truth. Although the raw input data sets are curated manually, class labels are derived algorithmically and may be noisy. For example, if an assignee holds only two patents belonging to different categories, we pick one of these two categories randomly as the assignee label. In other words, we do not model our confidence in the derived class labels. Results show that the proposed algorithms is robust to this lack of information.

Table II. Node labels and label priors of the *Patent* data set.

Id	Name	Prior probability
C_1	Chemical	0.2077
C_2	Computers & communications	0.07945
C_3	Drugs & Medical	0.0859
C_4	Electrical & Electronic	0.19292
C_5	Mechanical	0.434

6.2. Perturbation

We also used perturbed data sets to stress-test the methods. The advantage of such data is the ability to test the effectiveness with varying uncertainty level, and other sensitivity parameters. This provides a better idea of the inherent variations of the performance. Perturbed data sets are generated by either adding noisy edges or by removing existing edges to and from the real data sets. Noisy edges are new edges with low probability. The edge probability is sampled from a normal distribution $N(0, \sigma)$ in the interval $(0, 1]$. The parameter σ controls the probability standard deviation. As it gets larger the average edge probability increases, eventually interfering with edges in the real data sets. The parameter ϕ controls the ratio of noisy edges. Given the edge set A of a real data set, the number of added noisy edges is $\phi \cdot |A|$.

The existing edges to be removed are selected by sampling the edge set A uniformly at random. Existing edges are removed *after* adding noisy edges. The parameter Φ controls the ratio of edges to be removed. Given the edge set A of a perturbed data set, the number of retained edges is $(1 - \Phi)|A|$. The selection criterion is also known as *probability sampling*.

The existing labeled nodes to be unlabeled are selected by sampling the node set N randomly. The parameter Γ controls the ratio of labeled nodes, whose label is to be removed. Given the node set N of a real data set, the number of labeled nodes whose label is removed is $(1 - \Gamma)|N|$.

Unless otherwise specified, we used the following default perturbation parameters. The ratio of noisy edges (ϕ) is 3 and the standard deviation of noisy edges (σ) is 0.25. By default, we do not remove any edges or labels. Thus, Φ equals zero, and the ratio of known labels for the data sets are those reported in Section 6.1.

6.3. Evaluation Methodology

The accuracy is assessed by using repeated random sub-sampling validation. We randomly partition the nodes into training and validation subsets which are denoted by

N_T and N_V respectively. We use $2/3$ of the labeled nodes for training, and the remaining $1/3$ for validation. Even if $2/3$ may appear as a large fraction, note that it refers to the labeled nodes in the ground truth (that is rather limited).

For each method, we compute the confusion matrix M on the N_V set, where M_{ij} is the count of nodes labeled as i in the ground-truth that are labeled as j. Accuracy is defined as the ratio of true positives for all class labels:

$$Accuracy = \frac{1}{|N_V|} \sum M_{ii} \tag{3}$$

If all nodes are labeled correctly, M is a diagonal matrix. The experiment is repeated several times to get statistically significant results.

In all experiments we use the following parameters for the *uBayes+* algorithm. The ratio nodes of the sampled uncertain network (α) is 0.2. Among the sampled nodes, the ratio nodes used for training (β) is 0.1. In order to identify θ^*, the algorithm varies θ between 0.05 and 1 in 20 steps. These values were determined experimentally, and are the same for both datasets (the performance of the algorithms remains stable for small variations of these parameters).

We compared our techniques to two algorithms, which are the *wvRN* [39] and *Sampling* methods. The *wvRN* method is based on major weighted voting while the *sampling* technique is the average of major voting of possible world samples obtained from the uncertain network graph. Since these algorithms trade accuracy for running time, we limited the running time of these two algorithms to the time spent by the *uBayes* method. The *wvRN* method estimated the probability of node i to have label j as the weighted sum of class membership probabilities of neighboring nodes for label j. Thus, it works with a weighted deterministic representation of the network, where the edge probabilities are used as weights. Relaxation labeling is then used for inference. We additionally consider a version of the *wvRN* algorithm, *wvRN-20*, that is not time-bounded, but is bound to terminate after 20 iterations in the label relaxation procedure. As we discuss later, the accuracy of *wvRN* converges quickly and does not improve further after 20 iterations for both data sets.

The sampling algorithm samples networks in order to create deterministic representations. For each sampled instantiation, the *RN* algorithm [39] is used. In our experiments, we used 100 samples, since no significant improvement in the results was observed when using more samples (we tested the algorithm with up to 2000 samples). Note that links in sampled instantiations either exist or do not exist, and link weights are set to 1. This algorithm estimates class membership probabilities by voting on the different labelings over different instantiations of the network. The class with the largest vote is reported as the relevant label.

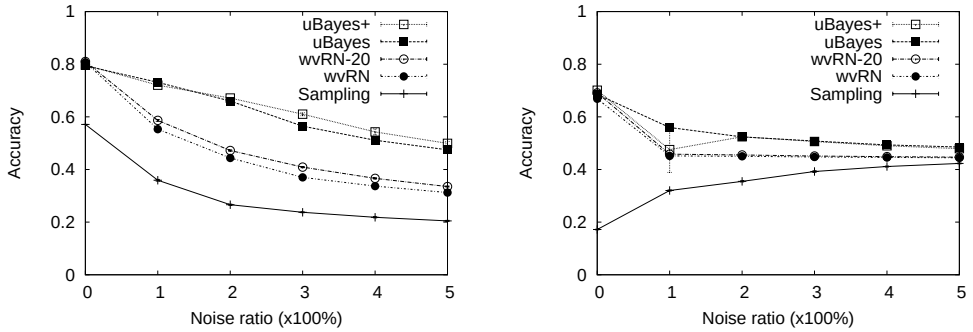
6.4. Classification Quality Results

In this section, we report our results on accuracy under a variety of settings using both real and perturbed data sets. The first experiment shows the accuracy by varying the ratio of noisy edges (ϕ) for the algorithms *uBayes*, *uBayes+*, *wvRN*, *wvRN-20* and *Sampling*. The results for the *DBLP* and *Patent* data sets are reported in Figures 3(a) and 3(b), respectively. The *Sampling* algorithm is the worst performer on both data sets, followed by *wvRN* and *wvRN-20*. On the *DBLP* data set, the accuracy of *wvRN-20* is slightly higher than that of *wvRN*. The *uBayes* and *uBayes+* algorithms are the best performers, with *uBayes+* achieving higher accuracy on the *DBLP* dataset when the ratio of noisy edges is above 200%. We observe that there is nearly no difference among the *uBayes*, *uBayes+*, *wvRN* and *wvRN-20* algorithms on both datasets when $\phi = 0$, while the percentage improvement in accuracy from *wvRN-20* to *uBayes+* when $\phi = 5$ (500%) is up to 49% for *DBLP* and 7% for *Patent*. It is worth noting that, as the

ratio of noisy edges increases, the accuracy for *Sampling* increases in the Patent data set. This is due to the high probability of label C_5 (0.434), as reported in Table II, which eventually dominates the process.

In the next experiment, we varied the standard deviation of the probability of the noisy edges (σ) for algorithms *uBayes*, *uBayes+*, *wvRN*, *wvRN-20* and *Sampling*. The results for the *DBLP* and the *Patent* data sets are reported in Figures 4(a) and 4(b), respectively. The *Sampling* algorithm again does not perform well, followed by the *wvRN* and *wvRN-20* algorithms. The *uBayes+* algorithm is consistently the best performer on the *DBLP* dataset, while there is nearly no difference between the *uBayes+* and *uBayes* algorithms on the *Patent* data set. The higher accuracy of *uBayes* and *uBayes+* is explained by their ability to better capture correlations between different class labels, a useful feature when processing noisy data sets. The better performance of *uBayes+* is due to its ability to ignore noisy labels that contribute negatively to the overall classification process. *uBayes+* is more accurate than *wvRN-20* with a percentage improvement up to 83% in the *DBLP* data set and 10% in the *Patent* data set, which represents a significant advantage.

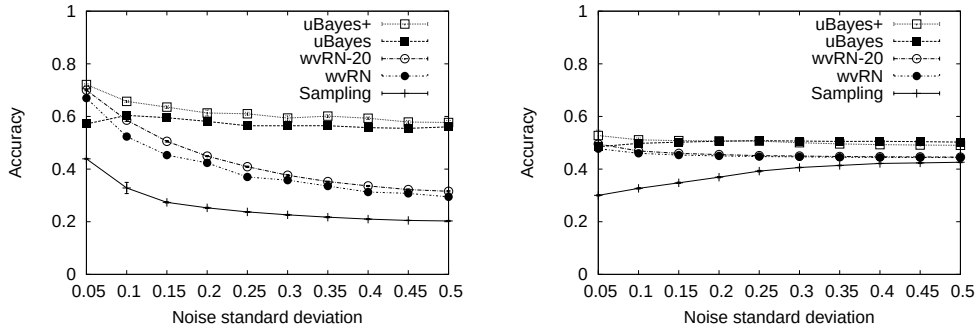
In the following experiment, we evaluate the accuracy when varying the ratio of labeled nodes (Γ) for algorithms *uBayes*, *uBayes+*, *wvRN*, *wvRN-20* and *Sampling*. (Default perturbation parameters are considered for the retained edges.) The results for the *DBLP* and *Patent* data sets are reported in Figures 5(a) and 5(b) respectively. In the *DBLP* dataset, the *wvRN* algorithm performs better than *wvRN-20*, while there is virtually no difference on the *Patent* dataset. The *uBayes+* algorithm is consistently the best performer on the *DBLP* dataset, while it performs slightly worse than *uBayes* on the *Patent* data set when Γ is below 0.2 (20%). We observe that the percentage improvement of *uBayes+* over *wvRN-20* is 50% on the *DBLP* dataset and 11% on the *Patent* dataset. The *Sampling* algorithm exhibits the lowest accuracy.



(a) DBLP (b) Patent

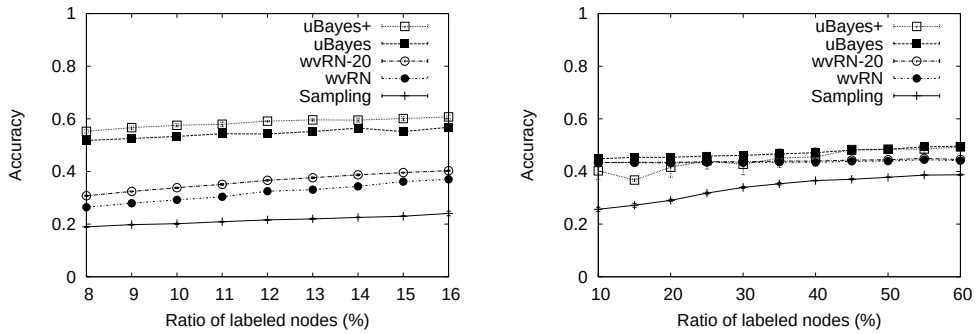
Fig. 3. **Accuracy with varying ratio of noisy edges for algorithms *uBayes*, *uBayes+*, *wvRN*, *wvRN-20* and *Sampling*.**

We now stress-test the proposed techniques by randomly removing a percentage of edges (Φ), as detailed in Section 6.2. The results for the *DBLP* and the *Patent* data sets are reported in Figures 6(a) and 6(b), respectively. *Sampling* consistently performs at the lower range, followed by *wvRN* and *wvRN-20*. *uBayes* and *uBayes+* perform consistently better on the *DBLP* dataset, with *uBayes+* performing poorly when the ratio of retained edges is below 60%. In this case, the resulting network is less connected, and the uncertain network sample used for the automatic parameter tuning becomes



(a) DBLP (b) Patent

Fig. 4. **Accuracy with varying standard deviation of probability of noisy edges for *uBayes*, *uBayes+*, *wvRN*, *wvRN-20* and *Sampling* algorithms.**

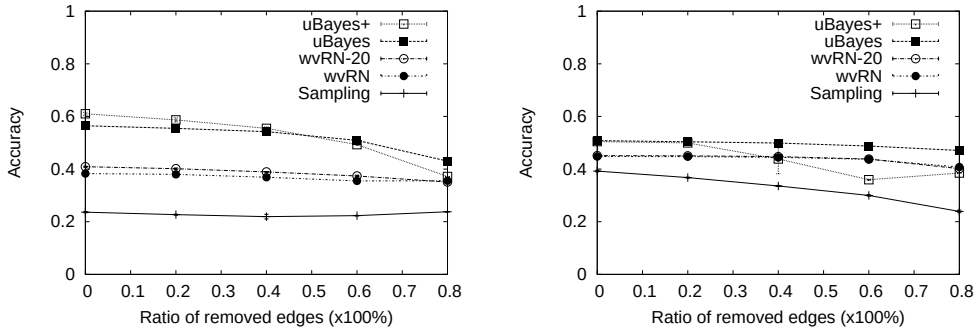


(a) DBLP (b) Patent

Fig. 5. **Accuracy with varying ratio of labeled nodes for *uBayes*, *uBayes+*, *wvRN*, *wvRN-20* and *Sampling* algorithms.**

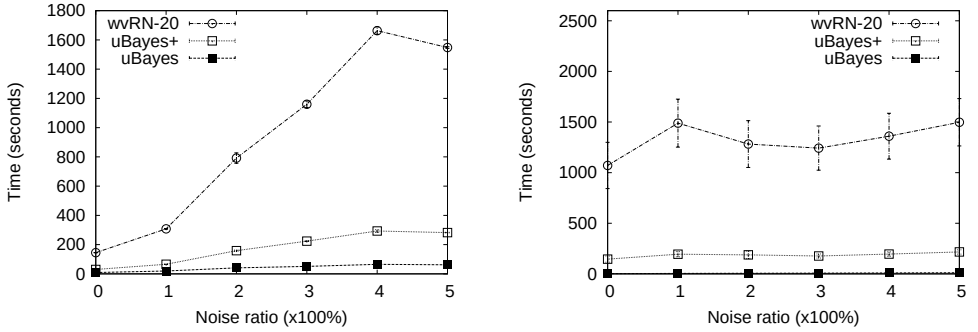
less robust to noisy conditions. In the *DBLP* dataset, the percentage improvement of *uBayes+* over *wvRN-20* is up to 49%.

In Table III, we report the confusion matrices for the *DBLP* and *Patent* data sets for the *uBayes+* algorithm. The confusion matrix provides some interesting insights, especially for cases where nodes were misclassified. Cell i, j reports the number of nodes with ground truth label C_i classified with label j. We observe that C_4, C_9 and C_{10} labels (networking, machine-learning and bioinformatics) in the *DBLP* data set lead to many misclassifications. This can be explained by the fact that these are the most frequent labels in the network (refer to Table I), and therefore have a higher probability of being selected. We also observe that misclassifications convey interesting and useful information. For example, excluding the C_4, C_9 and C_{10} classes, most of the misclassifications for class "Data Mining" are due to the "Information Retrieval" class, and vice versa. This points to the fact that the two communities are related to each other. Similar observations can be made on the *Patent* data set. For example, the "Chemical" and "Drugs & Medical" classes overlap, and show corresponding behavior in the confusion matrices.



(a) DBLP (b) Patent

Fig. 6. **Accuracy with varying ratio of retained edges for *uBayes*, *uBayes+*, *wvRN*, *wvRN-20* and *Sampling* algorithms.**



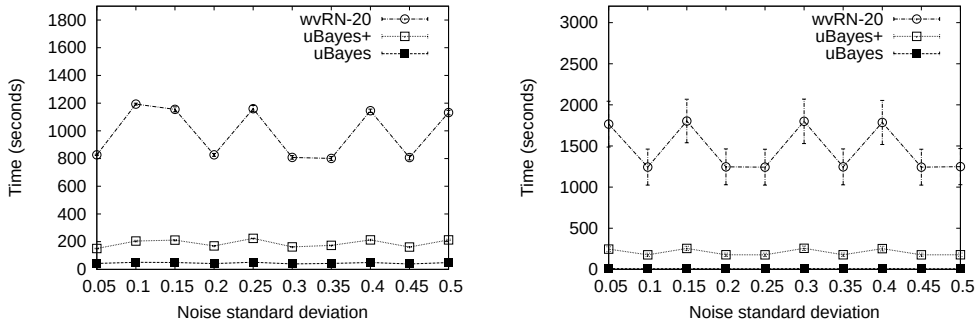
(a) DBLP (b) Patent

Fig. 7. **Time performance with varying ratio of noisy edges for *uBayes*, *uBayes+* and *wvRN-20* algorithms.**

Table III. Confusion matrix for *DBLP* dataset. True positives are indicated in bold.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}
C_1	**1943**	7	9	116	19	58	11	38	364	595	32	35	4	7
C_2	3	**620**	30	21	12	7	37	3	125	131	6	4	3	1
C_3	15	49	**1088**	51	25	18	13	3	489	261	11	8	11	3
C_4	55	21	22	**3907**	70	161	38	19	915	1076	69	87	23	8
C_5	12	17	26	182	**2237**	92	49	45	960	791	60	79	126	37
C_6	60	26	12	311	86	**1482**	26	39	423	464	105	80	35	7
C_7	7	36	13	43	14	12	**1396**	20	463	584	4	15	39	15
C_8	7	3	0	18	10	15	18	**619**	113	128	4	8	7	4
C_9	37	46	271	279	152	47	72	23	**5422**	1287	74	85	110	104
C_{10}	42	32	81	274	127	80	45	26	692	**8410**	79	58	44	45
C_{11}	28	10	7	135	22	168	9	13	258	321	**950**	46	5	3
C_{12}	27	13	18	236	62	70	33	26	428	483	35	**1230**	20	7
C_{13}	7	9	14	73	94	22	67	29	452	415	15	18	**873**	82
C_{14}	5	5	5	40	20	8	14	2	268	226	12	11	34	**756**

Finally, we report the accuracy of the *uBayes+RN* algorithm when varying the parameter δ between 0 and 1. Recall that δ controls the influence of the *RN* classifier on



(a) DBLP (b) Patent

Fig. 8. **Time performance with varying standard deviation of probability of noisy edges for *Bayes*, *Bayes+* and *wvRN-20* algorithms.**

Table IV. Confusion matrix for *Patent* data set. True positives are indicated in bold.

	C_1	C_2	C_3	C_4	C_5
C_1	**939**	5	34	79	3861
C_2	21	**328**	4	98	1421
C_3	181	0	**279**	26	1567
C_4	96	108	26	**574**	3798
C_5	292	63	26	133	**10020**

the overall classification process. In our experiments with both data sets, the accuracy of *uBayes+RN* was always slightly better than *uBayes+*, but never more than 5%. We observed nearly no difference among the different δ configurations. In the interest of space, we omit the detailed results.

We next provide some real examples of labeling results obtained with *uBayes+* on the Patent dataset. The "*Atari Inc.*" and "*Sega Enterprises, Ltd*" companies, which belong to the hall of fame of the video game industry, were not assigned to any category. Our algorithm correctly classified them as "Computers & Communications". Similarly, the companies "*North American Biologicals, Inc*" and "*Bio-Chem Valve, Inc*" were correctly labeled as "Drugs & Medical", since they are both involved in drug development and pharmaceutical research. Interestingly, "*Starbucks Corporation*" was labeled as "Chemical". Taking a close look at their patents, it turns out that a large fraction of them describe techniques for enhancing flavors and aromas that involve chemical procedures. Evidently, having labels for all the nodes in the graph allows for improved query answering and data analysis in general.

6.5. Efficiency Results

In this section, we assess running time efficiency on a variety of settings using both real and perturbed data sets. Figures 7(a) and 7(b) show the CPU time required by the algorithms when varying the ratio of noisy edges, for the *DBLP* and *Patent* data sets, respectively. Note that *Sampling* has the same time performance as *uBayes*. The *uBayes+* algorithm is nearly three times slower than *uBayes*. This is due to the automatic parameter tuning approach employed by the *uBayes+* algorithm We observe that the performance of *wvRN-20* almost always *considerably* worse than both *uBayes* and *uBayes+*. The same observation is true when we vary the standard deviation of the probability of the noisy edges (see Figures 8(a) and 8(b)). Note that the inference in

the *wvRN* algorithm is based on labeling relaxation, whose complexity is proportional to the size of the network and remains constant across iterations. On the contrary, the iterative labeling that *uBayes* and *uBayes+* use for their inference model becomes faster with each successive iteration, since it needs to visit a smaller part of the network. As the results show, the standard deviation does not affect the time performance of the algorithms. These experiments demonstrate that the two proposed algorithms effectively combine low running times with high accuracy and robustness levels.

In the final set of experiments, we evaluated the accuracy of all algorithms as a function of the time required for algorithmic execution by the baselines. Since the baselines tradeoff between running time and accuracy, it is natural to include the running time in the comparison process. In this case, we removed the constraint that *wvRN* and *Sampling* end their processing after a fixed amount of time or a specific number of iterations, and examined how their accuracy changes when the number of iterations (and consequently, processing time) increases. For reference, we also include the *uBayes* and *uBayes+* algorithms, which execute in a fixed amount of time. The results for the *DBLP* and *Patent* data sets are depicted in Figures 9 and 10 respectively. The graphs show that the accuracy of *wvRN* and *Sampling* is slightly increasing with time, but reaches an almost stable state after the first 10 iterations. (In our experiments, we stopped *wvRN* after 28 iterations in the *DBLP* data set and 16 iterations in the *Patent* data set, and the *Sampling* algorithm after 92 iterations in the *DBLP* data set and 28 iterations in the *Patent* dataset). Nevertheless, the *uBayes* and *uBayes+* algorithms achieve significantly better results in a much lower running time.

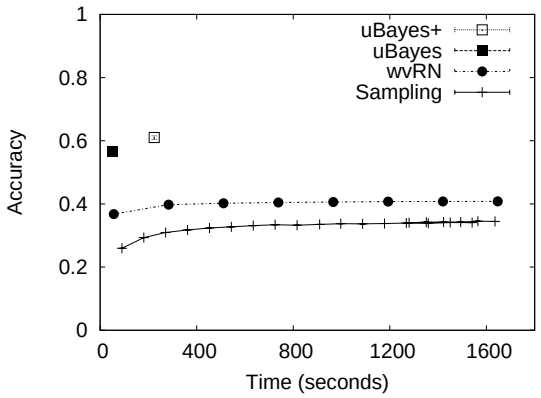


Fig. 9. **Accuracy with varying execution times for *uBayes*, *uBayes+*, *wvRN* and *Sampling* algorithms for the *DBLP* data set.**

7. CONCLUSIONS

Uncertain graphs are becoming increasingly popular in a wide variety of data domains. This is due to the statistical methods used to infer many networks, such as protein interaction networks and other link-prediction based methods. Consequently, the problem of collective classification has become particularly relevant for determining node properties in such networks.

In this paper, we formulate the collective classification problem for uncertain graphs, and describe effective and efficient solutions for this problem. To this effect, we describe an iterative probabilistic labeling method, based on the Bayes model, that treats uncertainty on the edges of the graph as first class citizens. In the proposed approach,

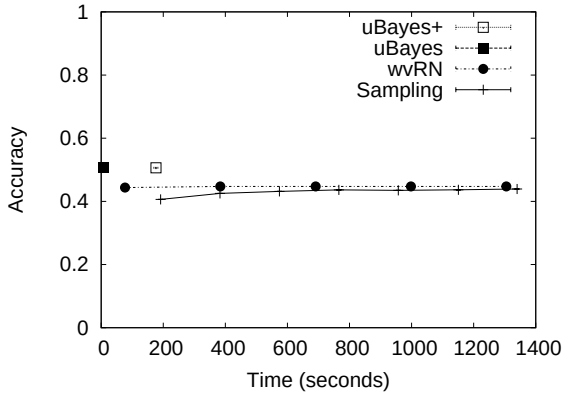


Fig. 10. **Accuracy with varying execution times for *uBayes*, *uBayes+*, *wvRN* and *Sampling* algorithms for the *Patent* data set.**

the uncertainty probabilities of the links are used directly in the labeling process. Furthermore, the methodology we describe allows for automatic parameter selection.

We have performed an experimental evaluation of the proposed approach using diverse, real-world datasets. The results show significant advantages of using such an approach for the classification process over more conventional methods, which do not directly use uncertainty probabilities.

Acknowledgments

Part of this work was supported by the FP7 EU IP project KAP (grant agreement no. 260111). Work of the second author was sponsored by the Army Research Laboratory under cooperative agreement number W911NF-09-2-0053.

8. REFERENCES

[1] Code and datasets for this paper available online at: *http://www.mi.parisdescartes.fr/%7Ethemisp/collectiveclassification/*, 2017.

[2] J. Leskovec and C. Faloutsos. Sampling from large graphs. KDD 2006.

[3] L. Eronen and H. Toivonen. Biomine: predicting links between biological entities using network models of heterogeneous databases. *BMC bioinformatics*, 13(1):119, 2012.

[4] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa. Injecting uncertainty in graphs for identity obfuscation. *Proceedings of the VLDB Endowment*, 5(11):1376–1387, 2012.

[5] Y. Yuan, G. Wang, L. Chen, and H. Wang. Efficient keyword search on uncertain graph data. *Knowledge and Data Engineering, IEEE Transactions on*, 25(12):2767–2779, 2013.

[6] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *International Conference on Very Large Data Bases (VLDB)*, pages 864–875, 2004.

[7] C. C. Aggarwal and P. S. Yu. A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 21(5):609–623, 2009.

[8] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. Proceedings of the Thirtieth international conference on Very large data bases, pp. 588–599, 2004.

[9] A. Fuxman, E. Fazli, and R. J. Miller. Conquer: Efficient management of inconsistent databases. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 155–166. ACM, 2005.

[10] Zhaonian Zou and Rong Zhu. Truss decomposition of uncertain graphs. *Knowl. Inf. Syst.*, 50(1):197-230, 2017.

[11] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. U. Nabar, T. Sugihara, and J. Widom. Trio: A system for data, uncertainty, and lineage. In *International Conference on Very Large Data Bases (VLDB)*, pages 1151–1154, 2006.

[12] L. Antova, C. Koch, and D. Olteanu. Query language support for incomplete information in the maybms system. In *International Conference on Very Large Data Bases (VLDB)*, pages 1422–1425, 2007.

[13] S. Singh, C. Mayfield, S. Mittal, S. Prabhakar, S. E. Hambrusch, and R. Shah. Orion 2.0: native support for uncertain data. In *ACM SIGMOD International Conference on Management of Data*, pages 1239–1242, 2008.

[14] S. Abiteboul, P. C. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. In *ACM SIGMOD International Conference on Management of Data*, pages 34–48, 1987.

[15] Rong Zhu and Zhaonian Zou and Jianzhong Li. SimRank computation on uncertain graphs. *32nd IEEE International Conference on Data Engineering (ICDE)* 2016.

[16] N. N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 1–12, 2007.

[17] Y. Gao. Shortest path problem with uncertain arc lengths. *Computers & Mathematics with Applications*, 62(6):2591–2600, 2011.

[18] J. Jestes, G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 23(12):1903–1917, 2011.

[19] C. Aggarwal, H. Wang. *Managing and Mining Graph Data*, Springer, 2010.

[20] C. Aggarwal. *Managing and Mining Uncertain Data*, Springer, 2009.

[21] S. Bhagat, G. Cormode, I. Rozenbaum. Applying link-based classification to label blogs, *WebKDD/SNA-KDD*, 2007.

[22] M. Bilgic, L. Getoor. Effective label acquisition for collective classification, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2008.

[23] R. Blake and P, Mangiameli. The effects and interactions of data quality and problem complexity on classification. *Journal of Data and Information Quality (JDIQ)*, 2(2), 8, 2011.

[24] P. Blau. *Inequality and heterogeneity: A primitive theory of social structure*. Free Press, NY, 1977.

[25] V. de Carvalho, W. Cohen, On the collective classification of email “speech acts”, *ACM Special Interest Group on Information Retrieval (SIGIR)*, 2005.

[26] M. Dallachiesa, C. Aggarwal, and T. Palpanas. Node classification in uncertain graphs, *SSDBM Conference*, 2014.

[27] M. Dash and A. Singhania. Mining in large noisy domains. *Journal of Data and Information Quality (JDIQ)*, 1(2), 8, 2008.

[28] B. Hall, A. Jaffe, M. Trajtenberg. The NBER patent citation data file: Lessons, insights and methodological tools, *National Bureau of Economic Research*, 2001.

[29] M. Hua, J. Pei. Probabilistic path queries in road networks: traffic uncertainty aware path selection. *International Conference on Extending Database Technology (EDBT)*, 2010.

[30] M. Ji, J. Han, M. Danilevsky. Ranking-based classification of heterogeneous information networks. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2011.

[31] G. Kollios, M. Potamias, E. Terzi. Clustering large probabilistic graphs. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 99, 2011.

[32] R. Jin, L. Liu, C. Aggarwal. Discovering Highly Reliable Subgraphs in Uncertain Graphs, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2011.

[33] R. Jin, L. Liu, B. Ding, H. Wang. Distance-constraint reachability computation in uncertain graphs. *International Conference on Very Large Data Bases (VLDB)*, 2011.

[34] X. Kong, P. Yu, X. Wang, A. Ragin. Discriminative Feature Selection for Uncertain Graph Classification. *SIAM International Conference on Data Mining (SDM)*, 2013.

[35] M. Ley, S. Dagstuhl. *DBLP Dataset*, http://dblp.uni-trier.de/xml/ August, 2012

[36] X. Li. A Bayesian approach for estimating and replacing missing categorical data. *Journal of Data and Information Quality (JDIQ)*, 1(1), 3, 2009.

[37] L. Liu, R. Jin, C. Aggrawal, Y. Shen. Reliable Clustering on Uncertain Graphs, *IEEE International Conference on Data Mining series (ICDM)*, 2012.

[38] Q. Lu, L. Getoor, Link-based classification, *International Conference on Machine Learning (ICML)*, 2003.

[39] S. Macskassy, F. Provost. A simple relational classifier. *Technical report*, 2003.

[40] M. Magnani and D. Montesi. A survey on uncertainty management in data integration. *Journal of Data and Information Quality (JDIQ)*, 2(1), 5, 2010.

[41] M. McPherson, L. Smith-Lovin, J. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

[42] M. Mezzanzanica, R. Boselli, M, Cesarini, and F. Mercorio. A Model-Based Approach for Developing Data Cleansing Solutions. *Journal of Data and Information Quality (JDIQ)*, 5(4), 13, 2015.

[43] O. Papapetrou, E. Ioannou, D. Skoutas. Efficient discovery of frequent subgraph patterns in uncertain graph databases. *International Conference on Extending Database Technology (EDBT)*, 2011.

[44] M. Potamias, F. Bonchi, A. Gionis, G. Kollios. k-nearest neighbors in uncertain graphs. *Proceedings of the Very Large Data Base Endowment (PVLDB)*, 2010.

[45] B. Taskar, P. Abbeel, D. Koller, Discriminative probabilistic models for relational data, *UAI*, 2002.

[46] Y. Yuan, G. Wang, H. Wang, L. Chen. Efficient subgraph search over large uncertain graphs. *International Conference on Very Large Data Bases*, 2011.

[47] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, *Neural Information Processing Systems (NIPS)*, 2004.

[48] D. Zhou, J. Huang, B. Schölkopf, Learning from labeled and unlabeled data on a directed graph, *International Conference on Machine Learning (ICML)*, 2005.

[49] X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, *ICML*, 2003.

[50] Z. Zou, H. Gao, J. Li. Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[51] Z. Zou, J. Li, H. Gao, S. Zhang. Finding top-k maximal cliques in an uncertain graph. *International Conference on Data Engineering (ICDE)*, 2010.

[52] J. Ren, S. D. Lee, X. Chen, B. Kao, R. Cheng, and D. Cheung. Naive bayes classification of uncertain data. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 944–949. IEEE, 2009.

[53] J. Dahlin and P. Svenson. A method for community detection in uncertain networks. In *Intelligence and Security Informatics Conference (EISIC), 2011 European*, pages 155–162. IEEE, 2011.