Clustering Large Probabilistic Graphs

George Kollios, Michalis Potamias, Evimaria Terzi.

Abstract—We study the problem of clustering probabilistic graphs. Similar to the problem of clustering standard graphs, probabilistic graph clustering has numerous applications, such as finding complexes in probabilistic protein-protein interaction networks and discovering groups of users in affiliation networks.

We extend the edit-distance based definition of graph clustering to probabilistic graphs. We establish a connection between our objective function and correlation clustering to propose practical approximation algorithms for our problem. A benefit of our approach is that our objective function is parameter-free. Therefore, the number of clusters is part of the output. We also develop methods for testing the statistical significance of the output clustering and study the case of noisy clusterings.

Using a real protein-protein interaction network and ground-truth data, we show that our methods discover the correct number of clusters and identify established protein relationships. Finally, we show the practicality of our techniques using a large social network of Yahoo! users consisting of one billion edges.

1 Introduction

Network data analytics lie in the core of many scientific fields such as social, biological and mobile ad-hoc networks. Typically, such network data are associated with uncertainty. This uncertainty is either due to the data-collection process or to machine-learning methods employed at preprocessing. Uncertainty may be also added to data for privacy-preserving reasons. We model such uncertain networks as *probabilistic graphs*. Every edge in a probabilistic graph is associated with a probability of existence [1], [2]. As an example, consider the probabilistic protein-protein interaction (PPI) networks. Edges in PPI networks represent interactions between proteins which result from combining error-prone measurements and therefore entail uncertainty [3].

We focus on the problem of partitioning a probabilistic graph into clusters. This is a fundamental problem for probabilistic graphs, just as it is for deterministic graphs. Partitioning a probabilistic graph into clusters has many applications such as finding complexes in protein-protein interaction networks and communities of users in social networks. A straightforward approach to clustering probabilistic graphs is to heuristically cast the probability of every edge into a weight and apply existing graph-clustering algorithms [4] on this weighted graph. This approach is problematic; not only there is no meaningful way to perform such a casting, but also there is no easy way to additionally encode normal weights on the edges.

In this paper, we present a principled approach to probabilistic graph clustering. Motivated by the *possible-world semantics* of probabilistic databases [5], [6] and probabilistic graphs [7], [8], we treat every probabilistic graph $\mathcal{G}$ as a generative model for deterministic graphs. Each such deterministic graph is a *possible world* of $\mathcal{G}$ and is associated with a probability to be generated.

Consider a deterministic graph $G = (V, E)$ and a partitioning, C, of the nodes in V. A *clustering objective function* $D(G, C)$ quantifies the cost of the clustering C with respect to G. The possible-world semantics dictate that the cost of a clustering C for a probabilistic graph $\mathcal{G}$ is the *expected value* of $D(G, C)$, over all possible worlds of $\mathcal{G}$ (i.e., $D(\mathcal{G}, C) = E[D(G, C)]$). Although this generalization is natural, it raises computational concerns. For instance, evaluating $D(\mathcal{G}, C)$ using the definition of expectation requires considering all, exponentially many, possible worlds of $\mathcal{G}$. Further, the expectation of well-established clustering objective functions (e.g., the maximum cluster diameter), is infinite since, typically, there exist possible worlds where parts of the graph are disconnected. Therefore, new definitions of the clustering problem in probabilistic graphs are necessary.

We view clustering C as a *cluster graph*, i.e., a graph consisting of disconnected cliques. Our optimization function is the *edit distance* between G and cluster graph C. In other words, $D(G, C)$ is the number of edges that we need to add and remove from G to get C. Given a probabilistic graph $\mathcal{G}$, we define pCLUSTEREDIT as the problem of finding the cluster graph C that has the minimum *expected* edit distance from $\mathcal{G}$. Our problem is a generalization of the CLUSTEREDIT problem introduced by Shamir et al. [9] for deterministic graphs.

Our framework for clustering probabilistic graphs has many desirable features. First, our objective function can be computed in polynomial time. That is, we avoid its evaluation for every possible world of $\mathcal{G}$. Further, the value of our objective function is never infinity; it is bounded by the number of node pairs in the graph. Also, the variance of our objective to *any* clustering depends solely on the graph and not on the specific clustering. This indicates that the edit distance is a robust objective

• *G. Kollios and E. Terzi are with the Computer Science Department, Boston University, Boston, MA. M. Potamias is with Smart Deals, Groupon, Palo Alto – this work was completed while the author was a PhD student at Boston University.*
E-mail: gkollios@cs.bu.edu, michalis@groupon.com, evimaria@cs.bu.edu

for clustering probabilistic graphs. Finally, our objective is parameter-free, hence, the number of clusters is part of the output.

Our contributions. We give a new definition of clustering in probabilistic graphs based on graph edit distance and we establish a connection between our problem and correlation clustering [10]. We exploit this connection in order to design efficient algorithms for clustering large probabilistic graphs. Our algorithms also provide approximation guarantees. Further, we establish a framework to compute deviations of a random world from the discovered clustering and to test the statistical significance of the resulting clusterings. Also, we study versions of our problem where the target clustering is itself noisy. Our experimental evaluation on a probabilistic protein-protein interaction network shows that our algorithms discover the correct number of clusters and identify known co-complex relationships among proteins. We also show that they can efficiently cluster a probabilistic social network from Yahoo! Groups with one billion edges.

Discussion: Our framework outputs a *partition* of the nodes of the probabilistic graph into groups. That is, it does not support probabilistic membership of nodes to clusters. Extending the proposed framework to find probabilistic assignments to clusters or identify overlapping clusters is material for future work.

Roadmap: The rest of the paper is organized as follows: After reviewing the related work in Section 2, we present our probabilistic-graph model in Section 3. We define the basic version of the pCLUSTEREDIT problem in Section 4 and give algorithms for solving it in Section 5. Some extensions to the basic pCLUSTEREDIT problem are presented in Section 6. In Section 7, we present a thorough experimental evaluation on real datasets. We conclude the paper in Section 8.

2 RELATED WORK

To the best of our knowledge, we are the first to define and study the problem of clustering probabilistic graphs using the possible-worlds semantics. However, uncertain data management and graph mining has motivated many studies in the data mining and database community. We highlight some of this work here.

Graph and probabilistic-graph mining: Clustering and partitioning of deterministic graphs has been an active area of research [11], [12], [13]. For an extensive survey on the topic see [4] and the references therein. Most of these algorithms can be used to handle probabilistic graphs, either by considering the edge probabilities as weights, or by setting a threshold value to the probabilities of the edges and ignoring any edge with probability below this threshold. The disadvantage of the first approach is that once probabilities are interpreted as weights, then no other weights can be taken into consideration (unless the probabilities are multiplied with edge weights – in which case this composite weight has no interpretation). The disadvantage of the second approach is that there is no principled way of deciding what the right value of the threshold is. Although both the above methodologies would result in an algorithm that would output some node clustering, this algorithm, contrary to ours, would not optimize an objective defined over all possible worlds of the input probabilistic graph.

Further, various graph mining problems have been studied recently assuming uncertain graphs [14], [15], [16], [17], [7], [8], [18]. For example, Hintsanen and Toivonen [15] looked at the problem of finding the most reliable subgraph, and Zou et al. [18] considered the problem of finding frequent subgraphs of an input probabilistic graph. More recently, Potamias et al. [7] proposed new robust distance functions between nodes in probabilistic graphs that extend shortest path distances from deterministic graphs and proposed methods to compute them efficiently. The problem of finding shortest paths in probabilistic graphs based on transportation networks has also been considered [19], [20]. The intersection between the above methods and ours is that all of them deal with probabilistic graphs. However, the graph-clustering task under the possible-worlds semantics has not yet been addressed by researchers in probabilistic graph mining.

Data Mining on Uncertain Data: Data mining over uncertain data has also received a lot of attention. Several classical data-mining problems have been revisited in the context of uncertainty. Examples include clustering of relational data [21], [22], [23], [24], [25], [26], [27], frequent-pattern mining [28], [29], [30], [18], and evaluating spatial queries [31]. All these works are tailored to model uncertain multi-dimensional data where uncertainty is associated either with the location of the data points in the space or with the actual existence of the data point in the dataset. One could think of defining probabilistic-graph clustering using the same ideas as those used for clustering uncertain multi-dimensional data. After all, the main idea there is to consider as an objective the expectation of standard clustering optimization criteria accross all possible worlds [22]. It may be tempting to try and use the same definitions for probabilistic graphs, particularly since standard clustering objectives (e.g., k-center or k-median) can be optimized in deterministic graphs. However, there is a fundamental difficulty with such clustering definitions in the probabilistic-graph setting: since there are many worlds where parts of the graph are disconnected, the distance (or proximity) of a node to any of the existing clustering centers can be infinity. Indeed, for non-trivial probabilistic graphs, there is always a non-zero probability of having a node with infinite distance to all the cluster centers. In that case, the optimization function becomes infinity. Therefore, new definitions of the clustering problem in probabilistic graphs are necessary. This paper addresses this challenge.

Probabilistic Databases: Probabilistic databases is another active research area, mostly focusing on the development of methods for storing, managing, and querying probabilistic data [32]. There exists fundamental work on the complexity of query evaluation on such data [6], on the computation of approximate answers to queries [33], [34] and on efficient evaluation of top-k queries [35], [36], [37], [38], [39]. Although we borrow the possible-world semantics pioneered by the probabilistic-database community, the computational problems we address here are different and require the development of new methodologies.

3 The probabilistic-graph model

Similar to deterministic graphs, probabilistic graphs may be undirected or directed and carry additional labels on the edges (such as weights). We focus on undirected probabilistic graphs. Our model assumes independence among edges.

We represent a probabilistic graph $\mathcal{G}$ using tuple $\mathcal{G} = (V, P, W)$; V corresponds to the set of nodes in $\mathcal{G}$, and we assume that $|V| = n$. P maps every pair of nodes to a real number in $[0, 1]$; P_{uv} represents the probability that edge $\{u, v\}$ exists. For weighted graphs we also use $W : V \times V \rightarrow \mathbb{R}$ to denote the weight associated with every edge $\{u, v\} \in V \times V$. In this paper we focus on unweighted probabilistic graphs. In this case, we represent the probabilistic graph as $\mathcal{G} = (V, P)$. For a probabilistic graph $\mathcal{G} = (V, P)$, we define its *complement* to be the probabilistic graph $\mathcal{G}' = (V, 1 - P)$.

One can think of a probabilistic graph as a generative model for deterministic graphs. A deterministic graph $G = (V, E_G)$ is generated by $\mathcal{G}$ by connecting two nodes u, v via an edge with probability P_{uv}. Deterministic graphs are an instance of probabilistic graphs for which $P_{uv} \in \{0, 1\}$. Also, $\mathcal{G}_{n,p}$ Erdős-Rényi random graphs [40] are an instance of probabilistic graphs where all edge probabilities are the same and equal to p.

We write $G \sqsubseteq \mathcal{G}$ to denote that G was generated by $\mathcal{G}$; the probability that $G = (V, E_G)$ is sampled from $\mathcal{G} = (V, P)$ is

$$\Pr[G] = \prod_{\{u,v\} \in E_G} P_{uv} \prod_{\{u,v\} \in (V \times V) \setminus E_G} (1 - P_{uv}). \quad (1)$$

If $M = \binom{|V|}{2}$, then there are 2^M distinct graphs that can be generated by $\mathcal{G}$. We use the term *possible world* to refer to each such graph.

4 Deterministic cluster graphs

In this section, we formulate the problem of clustering in probabilistic graphs as an optimization problem. First, we define the edit distance between two graphs. Then, we generalize this definition for probabilistic graphs and use it to define our graph-clustering problem.

*Definition 1 (*Edit Distance): Given two deterministic graphs $G = (V, E_G)$ and $Q = (V, E_Q)$ we define the *edit distance* between G and Q, to be the number of edges that need to be added or deleted from G in order to be transformed into Q. In other words,

$$D(G, Q) = |E_G \setminus E_Q| + |E_Q \setminus E_G|.$$

Let $\mathbf{G}$ (resp. $\mathbf{Q}$) denote the 0–1 adjacency matrix of G (resp. of Q). We have

$$D(G, Q) = \sum_{\substack{u=1,\\ v<u}}^{n} |\mathbf{G}(u, v) - \mathbf{Q}(u, v)|.$$

We extend this definition to the edit distance between a *probabilistic* graph $\mathcal{G} = (V, P)$ and a deterministic graph $Q = (V, E_Q)$. We define this distance, denoted by $D(\mathcal{G}, Q)$, to be the *expected* edit distance between deterministic graphs $G \sqsubseteq \mathcal{G}$ with the deterministic graph Q. This intuition is captured in the definition below.

Definition 2: The edit distance between a probabilistic graph $\mathcal{G}(V, P)$ and a deterministic graph $Q = (V, E_Q)$, is defined as the *expected* edit distance between every $G \sqsubseteq \mathcal{G}$ and Q. That is,

$$D(\mathcal{G}, Q) = \mathop{\mathbb{E}}_{G \sqsubseteq \mathcal{G}} [D(G, Q)] = \sum_{G \sqsubseteq \mathcal{G}} \Pr[G] D(G, Q). \quad (2)$$

Using Equation (2) we can compute $D(\mathcal{G}, C)$ by generating all 2^M possible worlds. For every world $G \sqsubseteq \mathcal{G}$, we can compute $\Pr[G]$ using Equation (1).This is an inefficient algorithm. However, the following holds.

Proposition 1: The expected edit distance between a probabilistic graph $\mathcal{G} = (V, P)$ and a deterministic graph $Q = (V, E_Q)$ in polynomial time.

Proof: Let $\mathbf{G}$ and $\mathbf{Q}$ denote the adjacency matrices of the deterministic graphs $G = (V, E_G)$ and $Q = (V, E_Q)$ respectively. Then we have

$$\begin{aligned} D(\mathcal{G}, Q) &= \mathop{\mathbb{E}}_{G \sqsubseteq \mathcal{G}} \left[\sum_{\substack{u=1\\ v<u}}^{n} |\mathbf{G}(u, v) - \mathbf{Q}(u, v)| \right] \\ &= \mathop{\mathbb{E}}_{G \sqsubseteq \mathcal{G}} \left[\sum_{\substack{u=1\\ v<u}}^{n} X_{uv} \right], \end{aligned}$$

where X_{uv} denotes the random variable $|\mathbf{G}(u, v) - \mathbf{Q}(u, v)|$. Note that X_{uv} takes values in $\{0, 1\}$: if $\mathbf{Q}(u, v) = 1$ (i.e., $\{u, v\} \in E_Q$), then $X_{uv} = 1$ with probability $1 - P_{uv}$ and $X_{uv} = 0$ with probability P_{uv}. On the other hand, if $\mathbf{Q}(u, v) = 0$ (i.e., $\{u, v\} \notin E_Q$), then $X_{uv} = 1$ with probability P_{uv} and $X_{uv} = 0$ with probability $1 - P_{uv}$. Using this observation and linearity of expectation we get

$$\begin{aligned} \mathop{\mathbb{E}}_{G \sqsubseteq \mathcal{G}} \left[\sum_{u<v} X_{uv} \right] &= \sum_{u<v} \left(\mathop{\mathbb{E}}_{G \sqsubseteq \mathcal{G}} X_{uv} \right) \\ &= \sum_{\{u,v\} \in E_Q} (1 - P_{uv}) + \sum_{\{u,v\} \notin E_Q} P_{uv}. \end{aligned} \quad (3)$$

The last expression is computable in time $O(m)$. □

We note that for weighted probabilistic graphs it becomes $\sum_{\{u,v\}\in E_Q}(1-P_{uv})W_{uv}+\sum_{\{u,v\}\notin E_Q}P_{uv}W_{uv}$. For the remainder of this paper we focus on unweighted graphs.

A central notion for the remainder of our analysis is the *cluster graph*. A cluster graph is a special deterministic graph that consists of vertex-disjoint disconnected cliques.

*Definition 3 (*CLUSTER GRAPH): A cluster graph $C = (V, E_C)$ is a deterministic graph with the following properties:

1) C defines a partition of the nodes in V into k parts, $V = \{V_1, \ldots, V_k\}$ such that $V_i \cap V_j = \emptyset$.
2) For *every* $i \in \{1, \ldots, k\}$ and for every pair of nodes $v \in V_i$ and $v' \in V_i$ we have that $\{v, v'\} \in E_C$.
3) For every $i, j \in \{1, \ldots, k\}$ with $i \neq j$ and every pair of nodes v, v' such that $v \in V_i$ and $v' \in V_j$, $\{v, v'\} \notin E_C$.

We can now define the following clustering problem:

*Problem 1 (*PCLUSTEREDIT*):* Given a probabilistic graph $\mathcal{G} = (V, P)$ find the cluster graph $C = (V, E_C)$ such that $D(\mathcal{G}, C)$ is minimized.

Note that the number of output clusters is not part of the input. In fact, the objective function itself dictates the number of clusters that are appropriate for every input.

The special case of the PCLUSTEREDIT problem, where the input graph is deterministic is called the CLUSTEREDIT problem. The CLUSTEREDIT problem was introduced by Shamir et al. [9] who also proved that CLUSTEREDIT is NP-hard. Therefore, PCLUSTEREDIT is also hard as a generalization of CLUSTEREDIT.

4.1 Deviations from expectation

The PCLUSTEREDIT problem focuses on finding the cluster graph C that minimizes the *expected* edit distance from the input probabilistic graph. Clearly, the value of $D(G, C)$ for different graphs $G \sqsubseteq \mathcal{G}$ may be different. How large are the observed differences in $D(G, C)$ across different worlds? 'Using the Chernoff bound, we show here that the mass of this distribution is concentrated around its mean.

Consider a probabilistic graph $\mathcal{G} = (V, P)$, a deterministic graph $G \sqsubseteq \mathcal{G}$ and any (not necessarily the optimal) cluster graph $C = (V, E_C)$. Also, let $\mathbf{G}$ and $\mathbf{C}$ be the 0–1 adjacency matrices of the graphs G and C respectively. Finally, for $u = 1 \ldots n$ and $v < u$ let $X_{uv} \stackrel{\triangle}{=} |\mathbf{G}(u, v) - \mathbf{C}(u, v)|$. Then, the expected edit distance between $\mathcal{G}$ and C is $D(\mathcal{G}, C) = \sum_{v<u} X_{uv}$. Each X_{uv} is a Bernoulli random variable: if $\{u, v\} \in E_C$, then $\Pr[X_{uv} = 1] = (1 - P_{uv})$. Else, if $\{u, v\} \notin E_C$, then $\Pr[X_{uv} = 1] = P_{uv}$. Therefore, the expected edit distance between $\mathcal{G}$ and C is a sum of random variables X_{uv}, for which probabilities $\Pr[X_{uv} = 1]$ are known. We can now apply the Chernoff bound (see e.g., [41]) such that for a cluster graph C, graph $G \sqsubseteq \mathcal{G}$ and $\delta > 0$,

$$\Pr\big[D(G, C) > (1+\delta)D(\mathcal{G}, C)\big] < \Big[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\Big]^{D(\mathcal{G},C)}, \quad (4)$$

and for any $\delta \in (0, 1]$

$$\Pr\big[D(G, C) < (1-\delta)D(\mathcal{G}, C)\big] < \Big[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\Big]^{D(\mathcal{G},C)}. \quad (5)$$

Inequalities (4) and (5) show that for any $G \sqsubseteq \mathcal{G}$, the probability that $D(G, C)$ deviates from $D(\mathcal{G}, C)$ is bounded. In practice, we use Inequality (4) as follows: Given a probabilistic graph $\mathcal{G}$, we first use one of our algorithms to obtain a cluster graph C that minimizes our objective, say at value D. Then, for any $D' < D$ (resp. $D' > D$) we can bound the probability that there exists a possible world G of $\mathcal{G}$, with edit distance from C that is smaller (resp. greater) than D'.

Variance: The variance of $D(G, C)$ for $G \sqsubseteq \mathcal{G}$ is bounded and independent of the clustering C:

$$\mathrm{Var}[D(G, C)] = \sum_{\{u,v\}\in V\times V, u<v} P_{uv}(1 - P_{uv}). \quad (6)$$

Proof: Using the same notation as above we have that

$$\mathrm{Var}[D(G, C)] = \mathrm{Var}\left[\sum_{u<v} X_{uv}\right] = \sum_{u<v} \mathrm{Var}[X_{uv}].$$

The second equality in the above equation holds because the X_{uv}'s are independent and therefore the variance of their sum is equal to the sum of their variances (see, e.g., [41]). Each one of the X_{uv}'s is also a Bernoulli random variable. If $\{u, v\} \in E_C$, then X_{uv} has mean $(1 - P_{uv})$ and variance $P_{uv}(1 - P_{uv})$. If $\{u, v\} \notin E_C$, then X_{uv} has mean P_{uv} and variance $P_{uv}(1 - P_{uv})$. Since the variance of X_{uv} is independent of whether $\{u, v\} \in E_C$, we have that

$$\sum_{u<v} \mathrm{Var}[X_{uv}] = \sum_{u<v} P_{uv}(1 - P_{uv}).$$

□

This indicates that the variance observed in the values of $D(G, C)$ does not depend on the clustering, but it depends only on the probabilistic graph itself. Probabilistic graphs that are very uncertain have high variance. The worst case is when for all pairs of nodes, $P_{uv} = 0.5$ (i.e., $\mathcal{G}$ is a random graph where every edge exists with probability 0.5). Then, the variance of $D(G, C)$ is the highest and equal to $0.25 * \binom{n}{2}$. In the special case that $\mathcal{G}$ is deterministic, the variance is equal to zero.

5 ALGORITHMS

One could solve the pCLUSTEREDIT problem using the following heuristic. First, decide on a threshold and ignore all edges that exist with probabilities below this threshold. Then, leverage an existing algorithm for clustering deterministic graphs to partition the points into clusters. Such an algorithm would have the following disadvantages: first, the choice of threshold seems arbitrary. Secongly, it will be very hard to prove that such an algorithm is an approximation algorithm for the pCLUSTEREDIT problem. In fact, it is not clear what is the objective that such an algorithm optimizes neither is it clear what the expectation of this objective over all possible worlds is.

Our algorithms for the pCLUSTEREDIT problem exploit the connection between our problem and the problems of *correlation clustering* [10] and *clustering aggregation* [42], [43]. Therefore, before describing the algorithms themselves we give a brief introduction to CORRELATIONCLUSTERING and highlight its connection to pCLUSTEREDIT.

The CORRELATIONCLUSTERING problem: In the CORRELATIONCLUSTERING problem [10] there are n objects in V and between any two pair of objects u, v there is a $(+)$ or a $(-)$ relation. We use E^+ (resp. E^-) to denote the set of pairs $u \neq v$ which are $(+)$-related (resp. $(-)$-related). The goal is to find a partition $\mathcal{P} = \{P_1, \ldots, P_m\}$ covering V and minimizing the number of disagreement pairs, i.e., $(+)$ pairs that are in different clusters and $(-)$ pairs that are in the same cluster. In the *weighted* CORRELATIONCLUSTERING instance, each pair $\{u, v\}$ is assigned two weights: $W^+_{uv} \geq 0$ and $W^-_{uv} \geq 0$. The cost of a clustering in this case is the sum of W^+_{uv} over all u, v that are in different clusters, plus the sum of W^-_{uv} over all u, v pairs that are in the same cluster. If $W^+_{uv} \in [0, 1]$ and $W^-_{uv} \in [0, 1]$ and for every pair u, v, $W^+_{uv} + W^-_{uv} = 1$, then we say that the correlation clustering weights satisfy the *probability constraint*. Since $W^+_{uv} = 1 - W^-_{uv}$, the objective function of the weighted CORRELATIONCLUSTERING with probability constraints can be expressed as follows:

$$\text{CC}(\mathcal{P}) = \sum_{\substack{(u,v) \\ \mathcal{P}(u)=\mathcal{P}(v)}} W^-_{uv} + \sum_{\substack{(u,v) \\ \mathcal{P}(u)\neq\mathcal{P}(v)}} \left(1 - W^-_{uv}\right). \quad (7)$$

Observe the similarity between the objective function of CORRELATIONCLUSTERING with probability constraints given in Equation (7) and the objective of the pCLUSTEREDIT problem as expressed in Equation (3). The two equations look identical except for the fact that we need to substitute $(1 - P_{uv})$ with W^-_{uv} and P_{uv} with $(1 - W^-_{uv})$. This observation hints that if there is a good approximation algorithm for the CORRELATIONCLUSTERING problem, then the same algorithm will be a good approximation algorithm for the pCLUSTEREDIT problem as well.

The `pKwikCluster` algorithm: Ailon et. al. [42] proposed the `KwikCluster` algorithm for the weighted CORRELATIONCLUSTERING problem. We adopt their algorithm for the pCLUSTEREDIT problem and we call it the `pKwikCluster` algorithm. The algorithm starts with a random node u and creates a cluster with all neighbors of u that are connected with u with probability higher than $1/2$. If no such node exists, u defines a singleton cluster. Having u and its cluster neighbors removed, the algorithm proceeds with the rest of the graph.

Given that `pKwikCluster` is a randomized expected 5-approximation algorithm for the CORRELATIONCLUSTERING problem with probability constraints [42], we have the following result.

Lemma 1: The `pKwikCluster` algorithm is a randomized expected 5-approximation algorithm for the pCLUSTEREDIT problem.

Running time: The worst-case time complexity of the `pKwikCluster` is $O(m)$. Thus, `pKwikCluster` is linear to the size of the input.

In fact, `pKwikCluster` is a randomized expected 5-approximation algorithm for the CLUSTEREDIT problem as well; this settles the approximability question of the CLUSTEREDIT problem posed by Shamir et. al. [9].

The `Furthest` algorithm: The `Furthest` algorithm uses the notion of *centers* to partition the probabilistic graph $\mathcal{G}$ in a top-down fashion. The algorithm starts by placing all nodes into a single cluster. Then, it finds the pair of nodes that have the smallest probability of having an edge between them. These two nodes become the centers of the two new clusters. The remaining nodes are assigned to the center with which it is more probable to share an edge.

This procedure is repeated iteratively. At each step we select as center the node that is not a center and has the maximum distance (i.e., minimum probability) to the current centers. The distance of a node from the current centers is defined as the maximum among the probabilities of having an edge with the current centers. The nodes are then assigned to the center to which they are connected with maximum probability. At the end of iteration i, the cluster graph C_i is formed which has cost $D(\mathcal{G}, C_i)$. If $D(\mathcal{G}, C_i) < D(\mathcal{G}, C_{i-1})$, then the algorithm continues to iteration $(i + 1)$. Otherwise, it stops and outputs C_{i-1}.

Running time: The `Furthest` algorithm needs to compute the distance between every node with the selected cluster centers. If k clusters are formed in the end, the worst-case running time of the `Furthest` algorithm is $O(nk^2)$.

The `Agglomerative` algorithm: The `Agglomerative` algorithm for pCLUSTEREDIT is a bottom-up procedure for the clustering problem. It starts by placing every node of the probabilistic graph $\mathcal{G} = (V, P)$ into a singleton cluster. At every iteration i, it constructs the cluster graph C_i that merges the pair of clusters that appeared

in C_{i-1} that have the largest average edge probability.[1] If the average probability of the closest pair of clusters is more than $1/2$, then the two clusters are merged into a single cluster. Otherwise the algorithm stops and outputs the current clustering.

Running time: A naive implementation of the `Agglomerative` algorithm requires $O(kn^2)$ time, where k is the number of clusters in the output. However, using a heap for retrieving the closest pair of clusters reduces the time complexity of the algorithm to $O(kn \log n)$.

5.1 Significance of clusterings

Application of the above algorithms on any probabilistic graph $\mathcal{G}$ will always output some cluster graph C. A question, very important to the practitioner, is whether this clustering reveals *any* significant cluster structure. Answering this question provides the practitioner with a measure of confidence (or lack thereof) on the output cluster graph C. We devise a randomization test that helps us decide whether the input probabilistic graph $\mathcal{G}$ is well-represented by the output cluster graph C.

Assume a probabilistic graph $\mathcal{G} = (V, P)$ and a cluster graph $C = (V, E_C)$. Consider a randomization method R, that given $\mathcal{G}$ and C, constructs a cluster graph $R(C) = C' = (V, E_C)$ by permuting the labels of the nodes in V. In this way, the clusters in C' have the same cardinality as the clusters in C, but the actual nodes that participate in each cluster are randomized.

Now, for the random clusterings C' constructed using the above random process R, we are interested in computing the following quantity:

$$\text{E-VAL}\,(\mathcal{G}, C, R) \quad = \quad \Pr\big[D(\mathcal{G}, C) \leq D(\mathcal{G}, R(C))\big]. \qquad (8)$$

We can compute the value of E-VAL $(\mathcal{G}, C, R)$ by creating random instances of $C' = R(C)$ and counting the fraction of instances in which the cost of the original cluster graph C is less than the cost of its randomized versions. The E-VAL of a clustering takes values in $[0, 1]$; the closer this value is to 1 the larger the confidence that clustering C is statistically significant.

5.2 Alternative clustering definitions

We briefly discuss some alternative clustering optimization functions.

CLUSTEREDIT **for the most-probable world:** This problem can be mapped to an unweighted instance of the CORRELATIONCLUSTERING. In particular, the most probable world G^* can be easily constructed by including all edges that have probability greater than 0.5. Therefore, our problem maps to CLUSTEREDIT on G^*. Thus, it can be solved by `KwikCluster`, which is a randomized expected 3-approximation algorithm. Note that even though the output of the approximation algorithm is the same as the output of `pKwikCluster` on $\mathcal{G}$, the optimal solutions of these problems may be different.

Most probable cluster graph: Notice that each cluster graph is a possible world. As such it has a probability of being generated by $\mathcal{G}$. A possible clustering definition i therefore to find the cluster graph with the maximum probability. This remains an open problem.

6 NOISY CLUSTER GRAPHS

In this section, we extend the PCLUSTEREDIT problem to allow noisy cluster graph outputs. We also show how we can exploit the machinery we developed in the previous sections to solve this problem.

So far, we have assumed that the output cluster graph is a deterministic graph consisting of disconnected cliques. A natural variant of this cluster graph is its noisy version, where some intracluster edges may be missing, and some intercluster edges may be present. We refer to such cluster graphs as *noisy cluster graphs*.

Accordingly, we define the (α, β)-cluster graph, denoted as $\mathcal{C}_{(\alpha,\beta)} = (V,\ P^{(\alpha,\beta)})$. $\mathcal{C}_{(\alpha,\beta)}$ defines a partition of the nodes in V into k parts $V_1, \ldots, V_k$ such that for any two nodes $v, v' \in V_i$, $P^{(\alpha,\beta)}_{vv'} = \alpha$; also for $v \in V_i$ and $v' \in V_j$ (with $i \neq j$) $P^{(\alpha,\beta)}_{vv'} = \beta$. A special case of (α, β)-cluster graph is the α-cluster graph (also denoted by $\mathcal{C}_\alpha$) defined for $\beta = 1 - \alpha$. For further intuition, consider α-cluster graphs with two clusters. If $\alpha = 1$ the 1-cluster graph consists of two disconnected cliques; if $\alpha = 0$ the 0-cluster graph is a fully connected bipartite graph.

Noisy cluster graphs are probabilistic graphs. Therefore, we define the edit distance between two probabilistic graphs.

Definition 4: The edit distance between two probabilistic graphs $\mathcal{G}(V, P)$ and $\mathcal{G}' = (V, P')$ defined over the same set of nodes, is the *expected* edit distance between every $G \sqsubseteq \mathcal{G}$ and $G' \sqsubseteq \mathcal{G}'$. That is,

$$\begin{aligned} D\left(\mathcal{G}, \mathcal{G}'\right) &= \mathop{\mathbb{E}}_{G \sqsubseteq \mathcal{G}, G' \sqsubseteq \mathcal{G}'} \left[D(G, G')\right] \\ &= \sum_{G \sqsubseteq \mathcal{G}, G' \sqsubseteq \mathcal{G}'} \Pr\left[G\right] \Pr\left[G'\right] D(G, G'). \end{aligned} \qquad (9)$$

Using observations similar to the ones we used in the proof of Proposition 1, we can show that the edit distance between two probabilistic graphs $\mathcal{G} = (V, P)$ and $\mathcal{G}' = (V, P')$ can be computed efficiently. The analytical form of this computation is:

$$D\left(\mathcal{G}, \mathcal{G}'\right) = \sum_{\substack{\{u,v\} \in V \times V \\ u < v}} \left(P_{uv}\left(1 - P'_{uv}\right) + \left(1 - P_{uv}\right) P'_{uv}\right). \qquad (10)$$

1. The average edge probability between two clusters V_i and V_j is defined as $\frac{1}{|V_i||V_j|} \sum_{u \in V_i, v \in V_j} P_{uv}$.

6.1 P2CLUSTEREDIT with α-cluster graphs

We can now generalize PCLUSTEREDIT to the following problem.

*Problem 2 (*P2CLUSTEREDIT*):* Given a probabilistic graph $\mathcal{G} = (V, P)$ and $\alpha \in [0, 1]$, find the α-cluster graph $\mathcal{C}_\alpha = (V, P^\alpha)$ such that $D(\mathcal{G}, \mathcal{C}_\alpha)$ is minimized.

This problem is NP-hard as a generalization of PCLUSTEREDIT. Nevertheless, the P2CLUSTEREDIT problem can be approximated at least as well as the PCLUSTEREDIT problem. In particular, we show that for any instance I_{P2C} of the P2CLUSTEREDIT problem we can construct an instance I_{PC} of the PCLUSTEREDIT problem such that the optimal solution of I_{PC} can be directly translated into the optimal solution of I_{P2C}. The following lemma formalizes the idea.

Lemma 2: If there is a γ-approximation algorithm for the PCLUSTEREDIT problem, then the same algorithm is a γ-approximation algorithm for the P2CLUSTEREDIT problem.

Proof: Let input probabilistic graph $\mathcal{G} = (V, P)$ and an α-cluster graph $\mathcal{C}_\alpha = (V, P^\alpha)$. Assume also that the α-cluster graph consists of a partitioning of the nodes into k parts $V_1, \ldots, V_k$. Let C_i be the set of pairs of nodes that belong in partition V_i. That is, $C_i = \{\{u, v\} \mid u \in V_i \text{ and } v \in V_i\}$. Finally, let $C = \cup_{i=1}^{k} C_i$ and $\widehat{C} = \{\{u, v\} \mid u \in V_i, v \in V_j \text{ and } i \neq j\}$. We compute the expected edit distance between $\mathcal{G}$ and $\mathcal{C}_\alpha$ using Equation (10). That is,

$$\begin{aligned} D(\mathcal{G}, \mathcal{C}_\alpha) &= \sum_{\{u,v\} \in C} \left((1-\alpha)P_{uv} + \alpha(1-P_{uv})\right) \\ &\quad + \sum_{\{u,v\} \in \widehat{C}} \alpha P_{uv} + (1-\alpha)(1-P_{uv}) \\ &= \sum_{\{u,v\} \in C} (\alpha + P_{uv} - 2\alpha P_{uv}) \\ &\quad + \sum_{\{u,v\} \in \widehat{C}} (1 - (\alpha + P_{uv} - 2\alpha P_{uv})). \end{aligned}$$

If we define

$$Y_{uv}(\alpha) \stackrel{\triangle}{=} (\alpha + P_{uv} - 2\alpha P_{uv}), \tag{11}$$

then we have that

$$D(\mathcal{G}, \mathcal{C}_\alpha) = \sum_{\{u,v\} \in C} Y_{uv}(\alpha) + \sum_{\{u,v\} \in \widehat{C}} (1 - Y_{uv}(\alpha)). \tag{12}$$

Equation (12) suggests that finding the optimal α-cluster graph for $\mathcal{G}$ is equivalent to finding the optimal (deterministic) cluster graph for the probabilistic graph $\mathcal{G}' = (V, Y_{uv}(\alpha))$. □

The consequences of Lemma 2 are apparent. Any of the algorithms proposed in Section 5 for solving the PCLUSTEREDIT problem, can also be used to solve P2CLUSTEREDIT. The only difference is that the algorithms will be applied to the probabilistic graph $\mathcal{G}' = (V, Y_{uv}(\alpha))$. Therefore ,the `pKwikCluster` is a randomized expected 5-approximation algorithm for P2CLUSTEREDIT.

Observe that the optimal 0-cluster graph of $\mathcal{G} = (V, P)$ is the optimal deterministic cluster graph for the complement of $\mathcal{G}$, i.e., for the graph $\mathcal{G}' = (V, 1 - P)$.

Deviations from expectation. Same as with deterministic cluster graphs, we can again use the Chernoff bound [41] to bound the deviation of $D(G, \mathcal{C}_\alpha)$ for any $G \sqsubseteq \mathcal{G}$, from the expected edit distance $D(\mathcal{G}, \mathcal{C}_\alpha)$. The expected edit distance $D(\mathcal{G}, \mathcal{C}_\alpha)$ is again a summation of independent Bernoulli variables. Therefore, bounds similar to those given by Equations (4) and (5) can be derived. The process is identical to the one we used in Section 4 and therefore we omit the details here. Similarly, we can use the analysis in Section 4 to compute the variance of the edit distance across possible worlds $G \sqsubseteq \mathcal{G}$. Using simple substitution, we have that $\text{Var}[G, \mathcal{C}_\alpha] = \sum_{u<v} Y_{uv}(\alpha)(1 - Y_{uv}(\alpha))$. In this case, the variance of the edit distance values does not depend only on the input probabilistic graph $\mathcal{G}$, but also on the parameter α.

6.2 Finding the value of parameter α

Until now, we assumed that the value of α is part of the input. What is the *optimal* value of the parameter α?

*Problem 3 (*MINA*):* Given a probabilistic graph $\mathcal{G}(V, P)$ find the value of α so that there exists an α-cluster graph $\mathcal{C}_\alpha = (V, P^\alpha)$ for which $D(\mathcal{G}, \mathcal{C}_\alpha)$ is minimized.

We have the following:

Lemma 3: The solution to the MINA problem is either $\alpha = 0$ or $\alpha = 1$.

The proof of this lemma is based on the linearity of the objective function $D(\mathcal{G}, \mathcal{C}_\alpha)$ with respect to α – see Equation (12). Clearly, the objective takes its minimum when α is set to its extreme values, i.e., 0 or 1.

Let us now illustrate how Lemma 3 can prove useful in practice. First, observe that a 0-cluster graph is a graph with no intracluster edges and with all the intercluster edges. Therefore, a 0-cluster graph is a good model for probabilistic graphs where non-edges define better clusters than edges (e.g., the two sides of a bipartite graph). A 1-cluster graph is a graph with no intercluster edges and all intracluster edges. The 1-cluster graph represents the mainstream definition of clusters in graphs: i.e., a cluster is a dense subgraph that is disconnected from the rest of the graph.

We can use the above intuition as follows: given a probabilistic graph $\mathcal{G}$, use any of the algorithms described above to solve the P2CLUSTEREDIT problem for $\alpha = 1$. Let $\mathcal{C}_1$ be the output 1-cluster graph, with cost $\mathbf{D}(\mathcal{G}) = D(\mathcal{G}, \mathcal{C}_1)$. Then, use the same algorithm to solve the P2CLUSTEREDIT problem for $\alpha = 0$. Let $\mathcal{C}_0$ be the output 0-cluster graph with cost $\overline{\mathbf{D}}(\mathcal{G}) = D(\mathcal{G}, \mathcal{C}_0)$. If $\mathbf{D}(\mathcal{G}) < \overline{\mathbf{D}}(\mathcal{G})$, then there is evidence of dense disconnected clusters in $\mathcal{G}$. Otherwise, the complement of $\mathcal{G}$, namely $\mathcal{G}'$, has greater cluster structure than $\mathcal{G}$.

6.3 Extension to (α, β)-cluster graphs

We extend the pCLUSTEREDIT problem to (α, β)-cluster graphs.

*Problem 4 (*p3CLUSTEREDIT*):* Given a probabilistic graph $\mathcal{G} = (V, P)$ and $\alpha, \beta \in [0, 1]$, find the (α, β)-cluster graph $\mathcal{C}_{(\alpha,\beta)} = (V, P^{(\alpha,\beta)})$ such that $D(\mathcal{G}, \mathcal{C}_{(\alpha,\beta)})$ is minimized.

Now assume a probabilistic graph $\mathcal{G} = (V, P)$ and an (α, β)-cluster graph $\mathcal{C}_{(\alpha,\beta)} = (V, P^{(\alpha,\beta)})$ that partitions V into $V_1, \ldots, V_k$. Similar to the proof of Lemma 2, we have that

$$D\left(\mathcal{G}, \mathcal{C}_{(\alpha,\beta)}\right) = \sum_{\{u,v\} \in C} \left((1-\alpha)P_{uv} + \alpha\left(1 - P_{uv}\right)\right) + \sum_{\{u,v\} \in \widehat{C}} (1-\beta)P_{uv} + \beta\left(1 - P_{uv}\right), \quad (13)$$

where C corresponds to the pairs of nodes where both the nodes of the pair belong in the same cluster; $\widehat{C}$ is used to denote the pairs of nodes where the two nodes of the pair belong in different clusters.

This equation suggests that solving Problem 4 for fixed α and β, boils down to finding a partitioning of the nodes in V so that Equation (13) is minimized. Unfortunately, for this version of the problem there is no direct transformation to pCLUSTEREDIT; i.e., there is no analogue of Lemma 2. Therefore, the approximability of Problem 4 remains an open problem.

Finding and interpreting the values of α and β. The analogue of the MINA problem for (α, β)-cluster graphs is to find the values of $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ such that, for input $\mathcal{G}$, there exists an (α, β)-cluster graph $\mathcal{C}_{(\alpha,\beta)}$ for which $D(\mathcal{G}, \mathcal{C}_{(\alpha,\beta)})$ is minimized. Observe that $D(\mathcal{G}, \mathcal{C}_{(\alpha,\beta)})$ (see Equation (13)) is a linear function with respect to α and β. Therefore, its minimum is one of the extreme values of (α, β) pairs, namely, $(0, 0)$, $(1, 0)$, $(0, 1)$ and $(1, 1)$.

Intuitively, $(0, 0)$-cluster graphs are good models for probabilistic graphs where there is very low probability to see *any* edge. Let S be the all-singleton cluster graph. Observe that S is among the optimal $(0, 0)$-cluster graphs for any $\mathcal{G}$. The edit distance of $\mathcal{G} = (V, P)$ from S is equal to $\mathbf{D}_S(\mathcal{G}) = D(\mathcal{G}, S) = \sum_{u<v} P_{uv}$.

Similarly, let Q be the cluster graph that puts all nodes in one cluster (i.e., clique). Q is among the optimal $(1, 1)$-cluster graphs for any $\mathcal{G}$ and the edit distance of $\mathcal{G}$ from Q is $\mathbf{D}_Q(\mathcal{G}) = D(\mathcal{G}, Q) = \sum_{u<v}(1 - P_{uv})$.

The optimal $(1, 0)$- and $(0, 1)$-cluster graphs are the $\mathcal{C}_1$ and $\mathcal{C}_0$ cluster graphs mentioned in the previous subsection with cost $\mathbf{D}(\mathcal{G})$ and $\overline{\mathbf{D}}(\mathcal{G})$ respectively.

Given a probabilistic graph $\mathcal{G}$, we call the four values $\mathbf{D}(\mathcal{G})$, $\overline{\mathbf{D}}(\mathcal{G})$, $\mathbf{D}_S(\mathcal{G})$ and $\mathbf{D}_Q(\mathcal{G})$ the *landmark* distances of $\mathcal{G}$. The landmark distances give good intuition about the structure of $\mathcal{G}$. For example, if $\mathbf{D}(\mathcal{G})$ is close to $\mathbf{D}_Q(\mathcal{G})$ (resp. $\mathbf{D}_S(\mathcal{G})$), then $\mathcal{G}$ does not have any other cluster structure than one large cluster (resp. n singleton clusters).

7 EXPERIMENTS

In this section, we present an experimental evaluation of our methodology. Our experiments verify that our algorithms output clusterings that are meaningful and statistically significant. We also demonstrate that our approximation algorithm `pKwikCluster` scales well and can handle real-world social networks with more than one billion edges. Finally, we show that `pKwikCluster` scales linearly in synthetic scale-free graphs. We ran all experiments on a Linux server with eight AMD Opteron processors with 64GB memory. All methods have been implemented in C++.

7.1 Protein-protein Interaction Network

The `CORE` dataset. For this experiment, we used the *core interaction network* provided by Krogan et al. ([3], Supplementary Table 7). The network consists of 2708 nodes that correspond to proteins. There are 7123 protein interactions that correspond to the network edges. The edge probabilities encode the confidence of the interaction between nodes. We refer to this network as `CORE` and we denote the underlying probabilistic graph as $\mathcal{G}_{\text{core}}$. There is no edge in the network with probability less than 0.27. Also, about 20% of the edges have probability greater than 0.98. The edge probabilities are uniformly spread in the remaining range $[0.27, 0.98]$. The dataset exhibits power-law degree distribution, short paths, and high clustering coefficient.

Quantitative performance of algorithms. The goal of this experiment is to compare the performance of our algorithms (`pKwikCluster`, `Agglomerative` and `Furthest`) with respect to the quality of the solutions they produce as well as their running times. The quality of a solution is measured in terms of our objective function, i.e., the expected edit distance of the output cluster graphs from the input probabilistic graph.

Apart from our algorithms, we also report results for three other methods: `pCast`, `Balls` and `MCL`. The `pCast` algorithm is a straightforward adaptation of the `Cast` heuristic [44] for probabilistic graphs. The `Balls` algorithm is a variant of `pKwikCluster`, proposed by Gionis et al. [43] for solving the clustering-aggregation problem. Finally, the `MCL` procedure is a two-parameter heuristic clustering technique based on random walks and matrix multiplications. Krogan et al. [3] report that they have manually tuned the two parameters of `MCL` to optimize the utility of the algorithm's results. We use this optimized output provided by Krogan et al ([3] Supplementary Table 10). We refer to this clustering as `Reference`.

We implemented `pKwikCluster` and `pCast`. For `Agglomerative`, `Furthest`, `Balls` we used the code provided by Gionis et al. [43], which is available online[2]. Both `pCast` and `Balls` require the setting of one parameter. For `pCast` we set the value of the parameter to

2. http://www.cs.helsinki.fi/hiit_bru/software/clustering_aggregation/

TABLE 1
CORE dataset: Summary of results in terms of edit distance and wallclock time of all algorithms.

Algorithm	Edit Distance	Wallclock Time (sec)
Reference	12230	n/a
pKwikCluster	4194	0.005
pCast	4502	100
Agglomerative	3420	10
Furthest	4612	150
Balls	4960	8

TABLE 2
CORE dataset: Summary of clustering results of all algorithms.

Algorithm	#clusters	TP	FP	FN
Reference	547	1791	11635	3589
pKwikCluster	491	838	2003	4542
pCast	1189	633	1338	4747
Agglomerative	543	946	1357	4434
Furthest	619	894	2322	4486
Balls	757	1120	1743	4260

0.7 after experimentally observing that it minimizes the expected edit distance. We set the parameter of Balls to its default value 0.17.

Table 1 summarizes the performance of all algorithms with respect to the expected edit distance of the solutions they produce from the input probabilistic graph $\mathcal{G}_{\text{core}}$. The running time of these algorithms for our implementations are also reported. Since pKwikCluster is a randomized algorithm, we ran it for 100 times and we report the best value of our objective achieved over these runs. In terms of edit distance, pKwikCluster and Agglomerative outperform the rest of the algorithms yielding solutions with edit distance 4194 and 3420 respectively. In terms of efficiency pKwikCluster is the fastest of all methods – its running time is linear to the number of edges in $\mathcal{G}_{\text{core}}$. In this small dataset, pKwikCluster takes just 5ms for a single run. Even though Agglomerative produces the highest quality clustering in CORE, its running time is 10s. Overall, the worst-case running time of Agglomerative is $O(kn \log n)$ and therefore, it cannot scale to more than a few thousands nodes. For the Reference clustering we do not report the running time since we utilize the reported results of MCL directly [3], without running the algorithm ourselves. However, MCL involves matrix multiplications so its complexity is at least quadratic to the number of nodes.

Qualitative performance of algorithms. In this experiment, we validate the output of our methods with respect to a known ground truth. Our results indicate that our techniques not only produce meaningful clusterings but also discover the correct number of clusters.

We use the MIPS database of protein complexes as ground truth [45]. MIPS complexes define co-complex relationships among proteins; a co-complex relationship is a pair of proteins that both belong to the same complex. Among the co-complex relationships present in MIPS, we only keep the ones that occur in $\mathcal{G}_{\text{core}}$. Thus, we end up with a ground truth of 5380 pairs of proteins. We emphasize that the complexes from MIPS correspond to overlapping sets of proteins; i.e., proteins may belong to more than one complexes. On the other hand, the output clusterings of the methods reported here are partitions, i.e., every protein participates in exactly one cluster.

Table 2 summarizes our results. The second column of this table reports the number of non-singleton clusters discovered by our techniques. Observe that our parameter-free techniques pKwikCluster, Agglomerative and Furthest discover between 490 and 619 non-singleton clusters, which is close to 547 discovered by Reference.

The rest of the columns of Table 2 summarize the confusion matrix constructed using the output of each algorithm. In particular, the third, fourth and fifth columns report respectively the number of True Positives (TP), False Positives (FP), and False Negatives (FN) attained by the different algorithms. All these calculations are done considering the MIPS database as the ground truth. The results indicate that our methods settle for a different trade-off than Reference. For instance, the number of TPs are 838 for pKwikCluster, while Reference discovers twice as many. On the other hand, the FPs of pKwikCluster are six times less than the ones of Reference.

To sum up, our techniques produce different clusterings compared to Reference. Yet, in terms of quality the results are comparable even though each clustering achieves different trade-offs. Besides, our algorithms optimize the edit distance between the probabilistic graph and the clustering without looking at the test set. On the other hand, Reference clustering has been produced by optimizing the parameters of MCL for the particular test-set [3].

Performance of pKwikCluster. Both previous experiments indicate that pKwikCluster, which is a provably approximation algorithm, performs very well in practice. The quality of the solutions it produces in terms of both their edit distance and their structural characteristics compare favorably to the other algorithms. Further, pKwikCluster scales well and is thus appropriate for large datasets. Therefore, we focus on pKwikCluster for the remainder of this paper.

Deviations from expectation. The goal of this experiment is to illustrate that in practice, the probability of sampling a random world with edit distance significantly larger than the expected edit distance optimized by pKwikCluster is negligible. Let C_{core} be the deterministic cluster graph output by the pKwikCluster algorithm for the CORE network. We now need to compute the probability that any random world $G \sqsubseteq \mathcal{G}_{\text{core}}$ has edit distance $D(G, C_{\text{core}})$ that is significantly different from $D(\mathcal{G}_{\text{core}}, C_{\text{core}})$. Instead of computing this probability exactly, we use Inequalities (4) and (5) to compute its

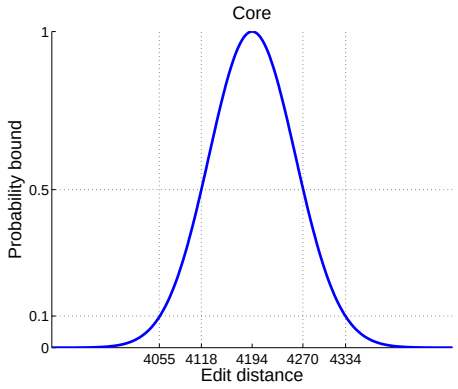

Fig. 1. CORE dataset: Probability bound for the distance between any random world and the output of pKwikCluster; x-axis: edit distance, y-axis: probability value.

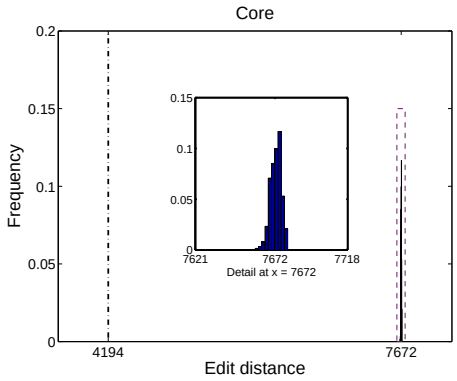

Fig. 2. CORE dataset: Distribution of the expected edit distance between the CORE network and the randomized versions of the output of pKwikCluster; x-axis: edit distance, y-axis: frequency. At 4194 we mark the distance of the original clustering.

bound. For the CORE dataset, we have $D(\mathcal{G}_{\text{core}}, C_{\text{core}}) = 4194$. Figure 1 shows that there is at most 0.1 probability to sample a world with distance less than 4055 or more than 4334. This exponential-tail bound illustrates that in practice a randomly sampled world will have a distance very close to the expected edit distance optimized by our objective.

In Section 4.1, we proved that the variance of the distance of a random world to any clustering is independent from the clustering. This variance can be computed analytically using Equation (6). The variance of CORE is 1096, which yields a standard deviation of 33.11.

Statistical Significance. We test the statistical significance of the clustering reported by pKwikCluster for the CORE network. We perform the randomization test R described in Section 5.1. If C_{core} is the cluster graph output by pKwikCluster for input $\mathcal{G}_{\text{core}}$, then we create randomized versions of C_{core} by maintaining the cluster sizes and permuting the labels of the nodes of $\mathcal{G}_{\text{core}}$. For 500 randomized instances of C_{core} created as above, we report that the value of E-VAL$(\mathcal{G}_{\text{core}}, C_{\text{core}}, R)$ is equal to 1. That is, the reported clustering C_{core} is statistically significant.

Figure 2 shows the value of $D(\mathcal{G}_{\text{core}}, C_{\text{core}})$ and a histogram of the values of $D(\mathcal{G}_{\text{core}}, R(C_{\text{core}}))$ for the randomized versions of C_{core}. The sample mean of $D(\mathcal{G}_{\text{core}}, R(C_{\text{core}}))$ is 7672 with a standard deviation as low as 3.4; this explains why the distribution resembles a spike. A simple calculation reveals that the original clustering is more than a thousand standard deviations away from the mean of the randomized clusterings. In practice, even if we were given the number of clusters and their cardinalities it would be impossible to find a clustering at least as good as the output of pKwikCluster by randomly permuting the node labels.

7.2 Affiliation Network

In this section, we focus on the application of pKwikCluster on clustering a large probabilistic social network with 1 billion edges. Our results demonstrate the scalability of pKwikCluster. We note that the rest of our methods cannot handle datasets as large as the one we consider in this part.

Dataset. We took the Yahoo! Groups user-group membership graph version 1.0 which is part of the Yahoo! Research Webscope datasets. The dataset corresponds to a small completely anonymized version of a bipartite graph showing membership patterns in Yahoo! Groups dating from early 2005. The dataset consists of 999,744 nodes, 638,124 groups, and 15,205,016 group memberships. The group sizes and user memberships exhibit power-law characteristics.

Starting from the information of users' memberships to groups, we construct the probabilistic Y!GROUPS network as follows: First we assign to every group g_i probability q_i. This probability encodes the probability of group g_i to induce a friendship link among any two of its members. Then, for any pair of users that co-participate in groups $g_1, \ldots, g_k$ we compute the probability of the edge between them as $1-(1-q_1)(1-q_2)\ldots(1-q_k)$. This is the probability that they are friends due to at least one of their common groups. This process uniquely defines a probabilistic graph. Using the recently proposed model by Lattanzi and Sivakumar [46] we set $q_i = |g_i|^{-a}$, for $a = 0.5$, noting that the value of a does not significantly change our results. Using this data-generation process we build the Y!GROUPS network that has a few more than $1B$ edges. We refer to this probabilistic graph as $\mathcal{G}_Y$. The edge probabilities in $\mathcal{G}_Y$ cover the whole $[0,1]$-range, most of them are close to 0. Indicatively, out of 1B edges, only about 15.5M of them have probability greater than 0.5.

Results for pKwikCluster. We ran pKwikCluster on the Y!GROUPS network 100 times. The best clustering C_Y reported by pKwikCluster has 110698 clusters and expected edit distance $D(\mathcal{G}_Y, C_Y) = 12.59 \times 10^6$. Each iteration of the algorithm on the Y!GROUPS dataset takes

TABLE 3
Landmark distances of the CORE and the Y!GROUPS networks

Landmark distances of $\mathcal{G}_{\text{core}}$	Landmark distances of $\mathcal{G}_Y$
$\mathbf{D}_Q(\mathcal{G}_{\text{core}}) = 3665 \cdot 10^3$	$\mathbf{D}_Q(\mathcal{G}_Y) = 499.74 \times 10^9$
$\mathbf{D}_S(\mathcal{G}_{\text{core}}) = 4839$	$\mathbf{D}_S(\mathcal{G}_Y) = 75.5 \times 10^6$
$\mathbf{D}(\mathcal{G}_{\text{core}}) = \mathbf{4194}$	$\mathbf{D}(\mathcal{G}_Y) = \mathbf{12.59 \times 10^6}$
$\overline{\mathbf{D}}(\mathcal{G}_{\text{core}}) = 4839$	$\overline{\mathbf{D}}(\mathcal{G}_Y) = 499.22 \times 10^9$

180 seconds.

Deviations from expectation and statistical significance. Our findings in this experiment are similar to the ones for the CORE network. The variance of the Y!GROUPS network is equal to 4.2×10^6, which yields a standard deviation around 2050. Therefore, the variance of random variable $D(G, C_Y)$ for any $G \sqsubseteq \mathcal{G}_Y$ is minuscule compared to the mean value (12.59×10^6). Running the randomization test R as before, we get E-VAL$(\mathcal{G}_Y, C_Y, R) = 1$. The average value of the expected edit distance obtained for the randomized clusterings is 15.1466×10^6, while the standard deviation of these values is as small as 160. The original output of pKwikCluster is thousands of standard deviations away from the sample mean distance of the randomized clusterings.

7.3 Scalability experiment on power-law graphs

Since most of the real-world graphs are scale-free, we test the scalability of our algorithm (pKwikCluster) using synthetically generated scale-free graphs.

We generate synthetic graphs using the Barabási-Albert (BA) model [47]. The BA graph-generation process adds nodes to the graph one at a time. Each new node is connected to k existing nodes with a probability that is proportional to the number of links that the existing nodes already have. We make these graphs probabilistic by generating a probability value uniformly at random in [0,1] for each edge. We call these graphs Probabilistic BA graphs.

Figure 3 shows the execution time of pKwikCluster in seconds as a function of the number of edges of the generated graph. For each data point shown in Figure 3, we create 20 probabilistic graphs and we run pKwikCluster 20 times on each graph. Thus, each point is the average of 400 executions. The execution time of pKwikCluster in seconds is shown on the y-axis. All graphs have 50000 nodes and the x-axis is the total number of edges in the graph. We vary the number of edges by choosing parameter k from $\{1, 2, \cdots, 10\}$. Figure 3 confirms that the running time of pKwikCluster scales linearly to the number of edges.

7.4 Noisy cluster graphs

(α, β)-clusters of CORE. The goal of this experiment is to gain further intuition on the cluster structure of the input graph $\mathcal{G}_{\text{core}}$ using the results of Section 6. We report the landmark distances of $\mathcal{G}_{\text{core}}$ in the first column of Table 3.

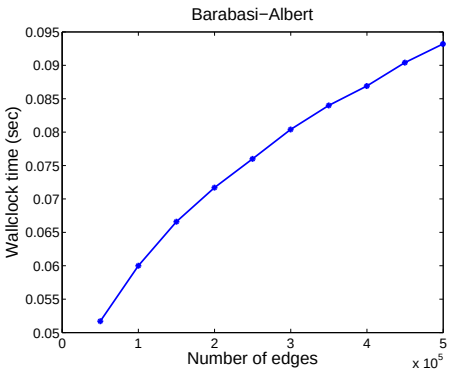

Fig. 3. Probabilistic BA graphs: Wallclock performance of pKwikCluster; x-axis: number of edges, y-axis: wallclock time in seconds.

Observe that $\mathbf{D}(\mathcal{G}_{\text{core}})$ has the minimum value among all four landmark distances. This is an indication that $\mathcal{G}_{\text{core}}$ has an inherent cluster structure and that $(1, 0)$-cluster graphs are good models for describing the clusters of the CORE network. We also evaluate the significance of the difference between $\mathbf{D}(\mathcal{G}_{\text{core}})$ and its closest landmark $\mathbf{D}_S(\mathcal{G}_{\text{core}})$. In particular, we compute the probability that a random world $G \sqsubseteq \mathcal{G}_{\text{core}}$ is closer in terms of edit distance to the all-singleton clustering than it is to the discovered $(1, 0)$-cluster graph. Using Inequality (4), we bound the probability of such an event (see Section 4.1 for details). The calculation yields a probability value less than 5.6×10^{-6}. In other words, for all except a negligible fraction of worlds, the obtained $(1, 0)$-cluster graph is the optimal model choice.

(α, β) clusters of Y!GROUPS. We consider the landmark distances of $\mathcal{G}_Y$ (see Table 3). Same as with CORE, $\mathbf{D}(\mathcal{G}_Y)$ has the minimum value among all four landmark distances. This indicates that $\mathcal{G}_Y$ has dense clusters and that $(1, 0)$-cluster graphs are better suited for describing the Y!GROUPS network. For a random choice of a world $G \sqsubseteq \mathcal{G}_Y$, the probability that the edit distance between G and the $(1, 0)$-cluster graph is smaller than the edit distance between G and the all-singleton clustering is less than 10^{-9}. Therefore, $(1, 0)$-cluster graphs are the best models for the Y!GROUPS network.

8 Conclusion

In this paper, we presented a thorough study of clustering probabilistic graphs using the edit distance metric. We focused on the problem of finding the cluster graph that minimizes the expected edit distance from the input probabilistic graph. Our formulation adheres to the possible-worlds semantics. Also, our objective function does not require the number of clusters as input; the optimal number of clusters is determined algorithmically.

We showed that our problem can be efficiently approximated, by establishing a connection with correlation clustering. In addition, we proposed various intuitive heuristics to address it. Further, we established a

framework to compute deviations of a random world to the proposed clustering and to test the significance of the resulting clusterings to randomized ones. Also, we addressed versions of our problem where the output clustering is itself noisy.

We tested our algorithms on a real probabilistic protein-protein interaction network and on a probabilistic social network that we created from Yahoo! Groups. Our experimental evaluation demonstrated that our algorithms not only produce meaningful clusterings with respect to established ground truth, but they also discover the correct number of clusters. Finally, we demonstrated that they scale to graphs with billions of nodes and that they produce statistically significant results.

9 Acknowledgments

George Kollios was partially supported by an NSF grant IIS-0812309. Evimaria Terzi was partially supported by NSF award #1017529, and gifts from Microsoft and Yahoo!.

References

[1] H. Frank, "Shortest paths in probabilistic graphs," *Operations Research*, vol. 17, no. 4, pp. 583–599, 1969.
[2] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM J. Comput.*, vol. 8, no. 3, pp. 410–421, 1979.
[3] N. J. Krogan, G. Cagney, and al., "Global landscape of protein complexes in the yeast saccharomyces cerevisiae," *Nature*, vol. 440, no. 7084, pp. 637–643, March 2006. [Online]. Available: http://dx.doi.org/10.1038/nature04670
[4] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.
[5] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom, "Uldbs: Databases with uncertainty and lineage," in *VLDB*, 2006, pp. 953–964.
[6] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," in *VLDB*, 2004.
[7] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, "k-nearest neighbors in uncertain graphs," *PVLDB*, vol. 3, no. 1, pp. 997–1008, 2010.
[8] Z. Zou, H. Gao, and J. Li, "Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics," in *KDD*, 2010, pp. 633–642.
[9] R. Shamir, R. Sharan, and D. Tsur, "Cluster graph modification problems," *Discrete Applied Mathematics*, vol. 144, no. 1-2, pp. 173–182, 2004.
[10] N. Bansal, A. Blum, and S. Chawla, "Correlation clustering," *Machine Learning*, vol. 56, no. 1-3, pp. 89–113, 2004.
[11] U. Brandes, M. Gaertler, and D. Wagner, "Engineering graph clustering: Models and experimental evaluation," *ACM Journal of Experimental Algorithmics*, vol. 12, 2007.
[12] G. Karypis and V. Kumar, "Parallel multilevel k-way partitioning for irregular graphs," *SIAM Review*, pp. 278–300, 1999.
[13] M. Newman, "Modularity and community structure in networks," *National Academy of Sciences*, vol. 103, pp. 8577–8582, 2006.
[14] Y. Emek, A. Korman, and Y. Shavitt, "Approximating the statistics of various properties in randomly weighted graphs," in *SODA*, 2011, pp. 1455–1467.
[15] P. Hintsanen and H. Toivonen, "Finding reliable subgraphs from large probabilistic graphs," *Data Min. Knowl. Discov.*, vol. 17, no. 1, pp. 3–23, 2008.
[16] X. Lian and L. Chen, "Efficient query answering in probabilistic rdf graphs," in *Proceedings of the 2011 international conference on Management of data*, ser. SIGMOD '11, 2011, pp. 157–168.
[17] O. Papapetrou, E. Ioannou, and D. Skoutas, "Efficient discovery of frequent subgraph patterns in uncertain graph databases," in *EDBT*, 2011, pp. 355–366.
[18] Z. Zou, J. Li, H. Gao, and S. Zhang, "Frequent subgraph pattern mining on uncertain graph data," in *CIKM*, 2009, pp. 583–592.
[19] M. Hua and J. Pei, "Probabilistic path queries in road networks: traffic uncertainty aware path selection," in *EDBT*, 2010, pp. 347–358.
[20] Y. Yuan, L. Chen, and G. Wang, "Efficiently answering probability threshold-based shortest path queries over uncertain graphs," in *DASFAA*, 2010, pp. 155–170.
[21] C. C. Aggarwal and P. S. Yu, "A framework for clustering uncertain data streams," in *ICDE*, 2008, pp. 150–159.
[22] G. Cormode and A. McGregor, "Approximation algorithms for clustering uncertain data," in *PODS*, 2008, pp. 191–200.
[23] S. Günnemann, H. Kremer, and T. Seidl, "Subspace clustering for uncertain data," in *SDM*, 2010, pp. 385–396.
[24] S. Guha and K. Munagala, "Exceeding expectations and clustering uncertain data," in *PODS*, 2009, pp. 269–278.
[25] B. Kao, S. D. Lee, F. K. F. Lee, D. W.-L. Cheung, and W.-S. Ho, "Clustering uncertain data using voronoi diagrams and r-tree index," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 9, pp. 1219–1233, 2010.
[26] H.-P. Kriegel and M. Pfeifle, "Density-based clustering of uncertain data," in *KDD*, 2005, pp. 672–677.
[27] W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip, "Efficient clustering of uncertain data," in *ICDM*, 2006, pp. 436–445.
[28] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Zuefle, "Probabilistic frequent itemset mining in uncertain databases," in *KDD*, 2009.
[29] L. Sun, R. Cheng, D. W. Cheung, and J. Cheng, "Mining uncertain data with probabilistic guarantees," in *KDD*, 2010, pp. 273–282.
[30] Q. Zhang, F. Li, and K. Yi, "Finding frequent items in probabilistic data," in *SIGMOD Conference*, 2008, pp. 819–832.
[31] M. L. Yiu, N. Mamoulis, X. Dai, Y. Tao, and M. Vaitis, "Efficient evaluation of probabilistic advanced spatial queries on existentially uncertain data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 1, pp. 108 –122, jan. 2009.
[32] N. N. Dalvi, C. Ré, and D. Suciu, "Probabilistic databases: diamonds in the dirt," *Commun. ACM*, vol. 52, no. 7, pp. 86–94, 2009.
[33] C. Koch, "Approximating predicates and expressive queries on probabilistic databases," in *PODS*, 2008.
[34] C. Re, N. N. Dalvi, and D. Suciu, "Efficient top-k query evaluation on probabilistic data," in *ICDE*, 2007.
[35] G. Cormode, F. Li, and K. Yi, "Semantics of ranking queries for probabilistic data and expected ranks," in *ICDE*, 2009.
[36] M. Hua, J. Pei, W. Zhang, and X. Lin, "Ranking queries on uncertain data: a probabilistic threshold approach," in *SIGMOD*, 2008.
[37] J. Li, B. Saha, and A. Deshpande, "A unified approach to ranking in probabilistic databases," *PVLDB*, vol. 2, no. 1, pp. 502–513, 2009.
[38] M. Soliman, I. Ilyas, and K. C.-C. Chang, "Top-k query processing in uncertain databases," in *ICDE*, 2007.
[39] K. Yi, F. Li, G. Kollios, and D. Srivastava, "Efficient processing of top-k queries in uncertain databases with x-relations," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 12, pp. 1669–1682, 2008.
[40] B. Bollobăs, *Random Graphs*. Cambridge University Press, 2001.
[41] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 2000.
[42] N. Ailon, M. Charikar, and A. Newman, "Aggregating inconsistent information: ranking and clustering," in *STOC*, 2005, pp. 684–693.
[43] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *TKDD*, vol. 1, no. 1, 2007.
[44] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering gene expression patterns," *Journal of Computational Biology*, vol. 6, no. 3/4, pp. 281–297, 1999.
[45] H. Mewes and al., "Mips: analysis and annotation of proteins from whole genomes," *Nucleic Acids Res*, vol. 32, pp. D41–44, 2004.
[46] S. Lattanzi and D. Sivakumar, "Affiliation networks," in *STOC*, 2009, pp. 427–434.
[47] Barabási, A. L. and Albert, R., "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

George Kollios is currently an Associate Professor in the Computer Science Department at Boston University in Boston, Massachusetts. He received his Diploma in Electrical and Computer Engineering in 1995 from the National Technical University of Athens,Greece; and the MSc and PhD degrees in Computer Science from Polytechnic University (now NYU-Poly), New York in 1998 and 2000 respectively. His research interests include spatio-temporal databases and data mining, database security, multimedia indexing, and approximation algorithms in data management.

Michalis Potamias received a Diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece in 2005, and the MA and PhD degrees in Computer Science from Boston University in 2008 and 2011 respectively. His research work has focused on uncertain data, large networks, and nearest neighbors indexing. Since August 2011 he has been a member of the Smartdeals team at Groupon.

Evimaria Terzi is an Assistant Professor in the Department of Computer Science at Boston Univeristy. She received a PhD in Computer Science from the University of Helsinki in 2007 and an MSc from Purdue University in 2002. Before joining Boston University in 2009, she was a Research Scientist at IBM Almaden Research Center. Her work focuses on algorithmic data mining, with emphasis on time–series and social–network analysis.