The Pursuit of a Good Possible World: Extracting Representative Instances of Uncertain Graphs

Panos Parchas[1] Francesco Gullo[2] Dimitris Papadias[1] Francesco Bonchi[2]

[1]Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{pparchas, dimitris}@cse.ust.hk

[2]Yahoo Labs
Barcelona, Spain
{gullo, bonchi}@yahoo-inc.com

ABSTRACT

Data in several applications can be represented as an uncertain graph, whose edges are labeled with a probability of existence. Exact query processing on uncertain graphs is prohibitive for most applications, as it involves evaluation over an exponential number of instantiations. Even approximate processing based on sampling is usually extremely expensive since it requires a vast number of samples to achieve reasonable quality guarantees. To overcome these problems, we propose algorithms for creating deterministic *representative* instances of uncertain graphs that maintain the underlying graph properties. Specifically, our algorithms aim at preserving the expected vertex degrees because they capture well the graph topology. Conventional processing techniques can then be applied on these instances to closely approximate the result on the uncertain graph. We experimentally demonstrate, with real and synthetic uncertain graphs, that indeed the representative instances can be used to answer, efficiently and accurately, queries based on several properties such as shortest path distance, clustering coefficient and betweenness centrality.

Categories and Subject Descriptors

E.1 [**Data Structures**]: Graphs and Networks; H.2.4 [**Systems**]: Query processing

Keywords

uncertain graph; possible world; representative

1. INTRODUCTION

Graphs constitute an expressive data representation paradigm used to describe entities (vertices) and their relationships (edges) in a wide range of applications. Sometimes the existence of the relationship between two entities is uncertain due to noisy measurements, inference and prediction models, or explicit manipulation. For instance, in biological networks, vertices represent genes and proteins, while edges correspond to *interactions* among them. Since these interactions are observed through noisy and error-prone experiments, each edge is associated with an uncertainty value [4]. In large social networks, uncertainty arises for various reasons [2]; the edge probability may denote the accuracy of a *link prediction* [21], or the influence of one person on another, e.g., in *viral marketing* [17]. Uncertainty can also be injected intentionally for obfuscating the identity of users for privacy reasons [6].

In all these applications the data can be modeled as an *uncertain graph* (also called probabilistic graph), whose edges are labeled with a probability of existence. This probability represents the confidence with which one believes that the relation corresponding to the edge holds in reality. Given the wide spectrum of application domains, querying and mining uncertain graphs has received considerable attention recently.

More formally let $\mathcal{G} = (V, E, p)$ be an uncertain graph, where $p : E \rightarrow (0, 1]$ is the function that assigns a probability of existence to each edge. Following the literature, we consider the edge probabilities independent [29, 15, 16] and we assume *possible-world* semantics [1, 9]. Specifically, the possible-world semantics interprets $\mathcal{G}$ as a set $\{G = (V, E_G)\}_{E_G \subseteq E}$ of $2^{|E|}$ possible deterministic graphs (worlds), each defined by a subset of E. The probability of observing any possible world $G = (V, E_G) \sqsubseteq \mathcal{G}$ is:

$$\Pr(G) = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)). \quad (1)$$

The exponential number of possible worlds usually renders exact query evaluation prohibitive. Indeed, even simple queries on deterministic graphs become #**P**-complete problems on uncertain graphs [32]. To better appreciate this fact, consider a binary predicate $q(G)$ in a deterministic graph G, e.g., q is a *reachability* query, which returns true if two input vertices are reachable from each other. In the context of uncertain graphs, the corresponding *reliability* query would ask for the probability of the vertices being reachable from each other. In general, the probability of predicate q in uncertain graphs is derived by the sum of probabilities of all possible worlds G for which $q(G) = true$:

$$q(\mathcal{G}) = \sum_{\substack{G \sqsubseteq \mathcal{G}, \\ q(G) = true}} \Pr(G).$$

As one cannot afford to materialize $2^{|E|}$ possible worlds, a common solution is to apply Monte-Carlo sampling, i.e.,

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.

http://dx.doi.org/10.1145/2588555.2593668.

to assess the query on a subset of randomly selected possible worlds. However, sampling is not always a viable option for large graphs because: i) sampling a possible world has a non-negligible cost as it requires generating a random number for each edge $e \in E$ and ii) a large number of samples is necessary to obtain a good approximation. Moreover, when the query output is complex (e.g., a data mining task producing a set of patterns) simple sampling is not sufficient: even if one is able to evaluate the query over a sample, it is not clear how to aggregate different samples.

In these cases the semantics of the analysis must be redefined for uncertain graphs, and new ad-hoc algorithms must be developed. However, designing new methods for data analysis problems is not always feasible. Organizations may have already invested in infrastructure (e.g., graph databases, graph processing software, etc.) for deterministic graphs, which they would wish to utilize, regardless of the uncertainty inherent in the data.

Motivated by the above, we aim at removing the uncertainty by producing representative instances of uncertain graphs. Queries can then be processed efficiently on the deterministic instance using conventional graph algorithms. In order to achieve accuracy, the representative instance should preserve the underlying structure of the uncertain graph. Starting from the observation that the vertex degree is one of the most fundamental properties of the structure of a graph, we conjecture that by preserving the degree of each vertex we capture the essence of the underlying uncertain graph, and thus accurately approximate other properties. The importance of vertex degrees in capturing structural graph properties has been well established in the literature of deterministic graphs, including communication network topologies [22] and complex network modeling [27].

We propose two methods for generating representative instances that capture the vertex degrees: Average Degree Rewiring (ADR) and Approximate B-matching (ABM). ADR involves two phases: first, it generates an instance with the same average vertex degree as the uncertain graph; then, it randomly rewires edges, if they lead to better approximation of the vertex degrees. ABM applies b-matching to obtain an initial instance, which then improves using weighted maximum bipartite matching.

Our extensive experimental evaluation on real and synthetic datasets confirms our conjecture: by preserving the vertex degrees, our representative instances can also capture several structural properties of the graph, including *shortest path distance*, *betweenness centrality* and *clustering coefficient*.

Summarizing, the contributions of the paper are:

1. We introduce the novel problem of extracting representative instances of uncertain graphs.
2. We propose ADR and ABM.
3. We experimentally demonstrate that the extracted representatives can accurately answer a variety of common graph processing tasks.

The rest of the paper is organized as follows. Section 2 overviews the related work. Section 3 defines the problem and presents two benchmark solutions. Sections 4 introduces the ADR algorithm, while Section 5 describes ABM. Section 6 contains an extensive experimental evaluation on real and synthetic datasets, and Section 7 concludes the paper.

2. BACKGROUND AND RELATED WORK

This section starts with an overview of previous work on uncertain graphs. Then, it presents research on node degree distribution due to its relevance to our problem. Finally, it describes *b-matching* because it is used in our ABM method.

2.1 Uncertain graphs

Uncertain relational databases have been well studied from different perspectives, such as SQL query evaluation, mining, ranking and top-k queries [3]. However, in many application domains, such as social, biological, and mobile networks, graphs serve as better models than relational tables. There have been three main directions of research on uncertain graphs: i) queries based on shortest-path distances, ii) pattern mining (frequent, reliable, etc.), and iii) subgraph (similarity) search.

Towards the first direction, Potamias *et al.* [29] study k-nearest neighbors proposing a set of alternative shortest path distance measures. They design methods to compute these distances by combining sampling with Dijkstra's algorithm. Similarly, based on the possible world semantics, Yuan *et al.* [34] return shortest-paths whose probability exceeds an input threshold. Jin *et al.* [16] propose two estimators to boost the accuracy of distance-constrained reachability queries, i.e, given two vertices s and t, return the probability that the distance from s to t is below a user-defined threshold.

In the second line of research, Zou *et al.* investigate mining frequent subgraphs [38] and top-k maximal cliques [37] in uncertain graphs. Jin *et al.* [15] study the problem of finding subgraphs, which are *highly reliable*, i.e., for which the probability of connectivity being maintained under uncertainty is larger than a given threshold.

In the third direction of research, Yuan *et al.* [36] propose a feature-based framework for subgraph search, while in [35] they study subgraph *similarity*. Finally, in a rather different kind of work [6], an uncertain graph is the output, rather than the input of the problem. Specifically, uncertainty is injected intentionally in a social graph for the sake of privacy, with the aim of achieving a form of identity obfuscation for the users.

2.2 Node degree distribution

According to [23], the degree distribution of a graph is the most frequently used topology characteristic. For instance, observations on the Internet's degree distribution had a huge impact on network topology research. Indeed, such studies on the importance of node degrees on the topology of deterministic graphs constitute our main motivation for using vertex degrees to capture the structure of uncertain graphs.

A sequence of non negative integers $d = \{d_1, d_2, \cdots, d_n\}$, with $d_1 \geq d_2 \geq \cdots \geq d_n$ is called *graphic*, if it is the degree sequence of some *simple* graph G. In such case, we say that G *realizes* the sequence d. Erdös and Gallai [10] describe the necessary and sufficient conditions for a sequence to be graphic. The theory on graphic degree sequences finds interesting applications in *complex network modeling* [22]. In order to generate topologies that capture the observed data, a common approach is to construct graphs that realize a predicted degree sequence of the complex network [13]. However, some structural characteristics (e.g. density, connectivity, etc.) depend heavily on the vertex processing order in [13]. Thus, several techniques use the output of [13]

as a seed, and take additional steps to *randomize* the graph [5], or induce specific characteristics, such as connectivity. These techniques perform local search by swapping edges in order to reach desired properties. We follow a similar approach in ADR .

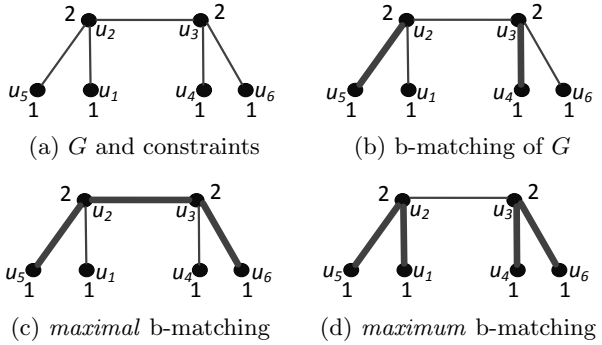
Another direction of research studies random processes on networks (the spread of epidemics [28], the evolution of social networks [33] etc.) by constructing *random graphs*. The popular Chung-Lu model [8] predicts the average distance of random graphs constructed with a given *expected* degree sequence, in order to justify facts such as the *small world phenomenon* [18]. Specifically, assuming a weight w_u for each vertex u, *any* edge (u, v) exists with probability $\rho = \frac{w_u w_v}{\sum_i w_i}$. On the other hand, in our uncertain graph model edge probabilities are given explicitly as an input to the problem, instead of being related to vertex weights. Moreover, the concept of *representative graph instance* does not apply to the other graph models. In the experimental section we use the Chung-Lu model to produce synthetic graphs.

2.3 b-Matching

Let an undirected graph $G = (V, E)$, and a set of capacity constraints $b(u)$: $V \to \mathbb{N}$. A subgraph $H = (V, E_H)$ of G is a *b-matching* of G, if the degree of each vertex $u \in V$ in H is at most $b(u)$. The term *b-matching* is used interchangeably to denote the subgraph H, or its edgeset E_H, depending on the context. If $b(u) = 1$ for all vertices of G, then *b-matching* reduces to the well known matching in graph theory. A b-matching is *maximal*, if the addition of any edge violates at least one capacity constraint. A *maximum b-matching* is a maximal b-matching with the largest number of edges.

Figure 1(a) shows an example graph, where the capacity constraint $b(u_i)$ is shown next to u_i. Figure 1(b) illustrates a b-matched graph of Figure 1(a), where the matched edges $((u_2, u_5), (u_3, u_4))$ are shown in bold. Since there is no violation of any capacity constraint, it is a valid b-matching. Figure 1(c) depicts a maximal b-matching: adding any other edge (e.g., (u_1, u_2)) would violate the capacity constraint of at least one vertex (e.g., u_2). Finally, Figure 1(d) illustrates the maximum b-matching.

Numerous exact and approximate solutions have been proposed for finding a maximum *b-matching* (see [14] for a survey). If the capacity constraints are bounded by a constant, the fastest exact algorithm is $O(|E|^{3/2})$ [25]. A greedy 1/2 approximation technique [14] solves the problem in $O(|E|)$. Several methods aim at weighted versions of the problem [24].



(a) G and constraints (b) b-matching of G

(c) *maximal* b-matching (d) *maximum* b-matching

Figure 1: b-matching example

3. PROBLEM DEFINITION AND BENCHMARK SOLUTIONS

Let $\mathcal{G} = (V, E, p)$ be an undirected uncertain graph, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $p : E \to (0, 1]$ is a function that assigns a probability of existence to each edge. For the sake of brevity, we denote the probability $p(e)$ of any edge $e \in E$ with p_e. We assume independent edge probabilities and possible world semantics, i.e., $\mathcal{G}$ is interpreted as a set $\{G = (V, E_G)\}_{E_G \subseteq E}$ of $2^{|E|}$ possible deterministic graphs (worlds), each defined by a subset of E.

Since most query processing tasks are prohibitively expensive for large uncertain graphs, we propose the extraction of a deterministic *representative instance* $G^* \sqsubseteq \mathcal{G}$ that captures the underlying properties of $\mathcal{G}$. Then, queries on the uncertain $\mathcal{G}$ can be efficiently processed using deterministic algorithms on G^*. In the following, we discuss desired properties of a representative instance, and introduce benchmark solutions for their generation. Table 1 contains the most common symbols throughout the paper.

Table 1: List of frequent symbols used in the paper.

Symbol	Definition
$\mathcal{G}$	uncertain graph
G	instance (possible graph) of $\mathcal{G}$
G^*	representative instance of $\mathcal{G}$
$deg_u(G)$	degree of vertex u in G
$[deg_u(\mathcal{G})]$	expected degree of vertex u in $\mathcal{G}$
$[deg(\mathcal{G})]$	expected average degree of $\mathcal{G}$
$dis_u(G)$	discrepancy of u in G, i.e., $deg_u(G) - [deg_u(\mathcal{G})]$
$\Delta(G)$	discrepancy of an instance G of $\mathcal{G}$
$\mathbf{P}$	sum of probabilities $\sum_{e \in E} p_e$

3.1 Representative instance

The representative instance G^* should conform well with the structural properties of $\mathcal{G}$ in *expectation*. A direct extraction of the "expected graph" from all possible worlds yielded by $\mathcal{G}$ is not easily achievable, as the definition of expected graph is intrinsically ill-posed. Indeed, the notion of expected value of any probability distribution needs i) an ordering among the points/objects of the domain of the distribution, and ii) a way of averaging (aggregating) among such objects. In our context the domain objects are graphs, hence it is not clear how to carry over either i) or ii).

Motivated by the above, we propose a criterion for extracting a representative instance that aims at preserving the *expected degree* of individual vertices. Particularly, our goal is to find a graph $G^* \sqsubseteq \mathcal{G}$ such that the degree of any vertex u in G^* is as close as possible to the *expected degree* of u in $\mathcal{G}$. Formally, given an uncertain graph $\mathcal{G} = (V, E, p)$ and a vertex $u \in V$, the *expected degree* of u in $\mathcal{G}$ is the summation of the probabilities of u's adjacent edges:

$$[deg_u(\mathcal{G})] = \sum_{e=(u,v) \in E} p_e$$

When the uncertain graph is implied, we write for convenience $[deg_u]$. Moreover, we define the *discrepancy* $dis_u(G)$ of a vertex u in an instance $G \sqsubseteq \mathcal{G}$ as the difference of $u's$ degree in G to its expected degree, i.e., $dis_u(G) = deg_u(G) - [deg_u]$. If the graph instance is implied, we equivalently write dis_u. Given the individual vertex discrepancies

dis_u, we define the overall discrepancy Δ of a possible graph G as follows:

DEFINITION 1. *Given an uncertain graph $\mathcal{G} = (V, E, p)$, the* discrepancy *of any possible graph $G \sqsubseteq \mathcal{G}$ is defined as*

$$\Delta(G) = \sum_{u \in V} |dis_u(G)| \tag{2}$$

The problem we tackle in this work is the following:

PROBLEM 1 (REPRESENTATIVE INSTANCE). *Given an uncertain graph $\mathcal{G} = (V, E, p)$, find a possible graph $G^* \sqsubseteq \mathcal{G}$ such that:*

$$G^* = \arg\min_{G \sqsubseteq \mathcal{G}} \Delta(G). \quad \square$$

Characterizing the complexity class of Problem 1 is non-trivial and represents an interesting open question. Our conjecture is that the problem is hard, or at least not solvable exactly in reasonable time for large graphs. To this purpose, note that Problem 1 can alternatively be formulated as an integer linear programming problem. Each edge $e \in E$ is assigned a binary variable $x_e = \{0, 1\}$, where $x_e = 1$ if and only if e is included in the result set. Then, the discrepancy of a vertex u in G can be expressed as $dis_u(G) = \sum_{e=(u,v) \in E}(x_e - p_e)$. Thus, Problem 1 becomes:

$$\begin{aligned} &\min |\mathbf{A}(\mathbf{x} - \mathbf{p})| \\ &\mathbf{x} = \{0, 1\}^{|E|} \end{aligned} \tag{3}$$

where $\mathbf{p} = (0, 1]^{|E|}$ is the vector containing the edge probabilities of the input uncertain graph $\mathcal{G}$ and $\mathbf{A} = \{0, 1\}^{|V| \times |E|}$ is the incidence matrix of $\mathcal{G}$. The formulation in (3) corresponds to a special case of the *closest vector* problem, which is known to be **NP**-hard [26]. Moreover, as discussed in Section 5, when *all* expected degrees are integers, Problem 1 can be solved by *b-matching* algorithms, among which the fastest runs in $O(|E|^{3/2})$ time [25]. Within this view, given that our main goal is to provide solutions that are scalable enough to deal with the large size of today's real-world graphs, we directly aim at approximate, yet efficient algorithms.

Representative instances vastly accelerate query processing on uncertain graphs because: i) they eliminate the overhead of generating a large number of samples and ii) the query is executed once (on the representative) instead of numerous times (for each sample). Consider for example a nearest-neighbor query. State-of-the-art approaches to this query type perform Dijkstra expansions on multiple samples [29]. Depending on the definition of the distance measure, expansion for some samples can be avoided or terminated early, when it cannot improve the current result. Nevertheless, the method has usually very high cost. On the other hand, the same query in our framework can be processed efficiently by applying any deterministic nearest neighbor algorithm on the representative G^*. As shown in our experimental evaluation, the representatives extracted by our algorithms indeed capture well the relevant properties of $\mathcal{G}$, in this case shortest path distances. Thus, the query on G^* is expected to return a good approximation of the nearest neighbor set.

3.2 Benchmark solutions

A straightforward way to generate a representative of an uncertain graph is to consider the instance with the highest probability to incur [29]. According to Equation 1, this *most probable* (MP) instance corresponds to the graph containing all the edges e that have probability $p_e \geq 0.5$. Since MP does not conform with any structural property of $\mathcal{G}$, it is expected to be a poor representative. As an example, if all edges in $\mathcal{G}$ have probability < 0.5, then MP is a graph with no edges.

An alternative benchmark, namely *Greedy Probability* (GP), aims at reducing the overall discrepancy using a greedy approach. Specifically, GP first sorts the edges of the graph in non-increasing order of their probabilities. Then, at each iteration, the algorithm considers an edge $e = (u, v)$ of the sorted list, and includes it to the result set, if $|dis_u + 1| + |dis_v + 1| < |dis_u| + |dis_v|$, i.e., the addition of e decreases the total discrepancy. The complexity of the algorithm is $O(|E| \log |V|)$ because it is dominated by the sorting step.

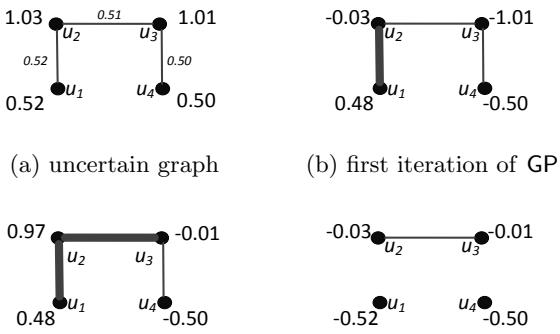
Algorithm 1 Greedy Probability GP

Input: uncertain graph $\mathcal{G} = (V, E, p)$
Output: approximate representative $G^* = (V, E^*)$
1: $E^* \leftarrow \emptyset$
2: sort E in non-increasing order of their probabilities
3: **for each** $e = (u, v) \in E$ **do**
4: **if** $|dis_u + 1| + |dis_v + 1| < |dis_u| + |dis_v|$ **then**
5: $E^* \leftarrow E^* \cup \{e\}$
6: $G^* \leftarrow (V, E^*)$

Algorithm 1 presents the pseudocode of GP. Intuitively, each edge (u, v) affects only the degrees of vertices u, v. Thus, if the condition in line 4 is satisfied, the value of Equation 2 decreases, leading to a better solution. Due to the initial sorting, the most probable edges are considered first. Such edges have large contribution to the expected degrees of the incident vertices, and at the same time they lead to a highly probable representative. However, due to its "blind" search strategy, GP may fail to find a good solution, as shown in the example of Figure 2.

Figure 2(a) illustrates an uncertain graph and the associated edge probabilities. Next to each vertex we include its expected degree, derived from the probabilities of adjacent edges. At the first iteration, GP picks the edge (u_1, u_2) with the highest probability and adds it to the result set E^*. Figure 2(b) shows E^* after the first iteration, where the expected degrees of the vertices have been replaced by their discrepancies. At the second iteration, GP considers edge (u_2, u_3). The inclusion of (u_2, u_3) in E^*, decreases the total discrepancy of u_2 and u_3 from 1.04 to 0.98 (see Figure 2(c); edges of E^* are bold). The final iteration, discards edge (u_3, u_4) since it would increase the discrepancy by 0.99. Thus, the representative returned by GP contains edges $\{(u_1, u_2), (u_2, u_3)\}$ with $\Delta = 0.48 + 0.97 + 0.01 + 0.50 = 1.96$.

The optimal solution shown in Figure 2(d), contains a single edge (u_2, u_3) with $\Delta = 0.52 + 0.03 + 0.01 + 0.50 = 1.06$. Note that for the same example, the representative produced by MP would include *all* edges since their probabilities are at least 0.5. This would yield an even worse solution than that of GP with $\Delta = 2.95$. To overcome the problems of the benchmark solutions, in the following sections we propose more sophisticated algorithms that aim at explicitly improving discrepancy through targeted search.



(a) uncertain graph (b) first iteration of GP

(c) second iteration of GP (d) optimal instance

Figure 2: GP example

4. AVERAGE DEGREE REWIRING (ADR)

Average Degree Rewiring (ADR) involves two phases: 1) it creates an instance $G_0 = (V, E_0)$ of the uncertain graph that preserves the average vertex degree and 2) it iteratively improves G_0 by rewiring, i.e., replacing edges in E_0, so that the total discrepancy is reduced.

The following lemma describes the efficient computation of the expected average degree for an uncertain graph.

LEMMA 1. *The expected average degree $[deg(\mathcal{G})]$ of an uncertain graph $\mathcal{G} = (V, E, p)$, is $[deg(\mathcal{G})] = \frac{2}{|V|}\mathbf{P}$, where $\mathbf{P}$ is the sum of all the edge probabilities in $\mathcal{G}$.*

PROOF. By definition, the average degree $deg(G)$ of a deterministic graph G is equal to $deg(G) = \frac{1}{|V|}\sum_{u\in V} deg_u$. Due to the linearity of expectation, the expected average degree is:

$$[deg(\mathcal{G})] = \left[\frac{1}{|V|}\cdot\sum_{u\in V} deg_u\right] = \frac{1}{|V|}\cdot\sum_{u\in V}[deg_u] =$$

$$= \frac{2}{|V|}\cdot\sum_{e\in E} p_e = \frac{2}{|V|}\cdot\mathbf{P} \quad \square$$

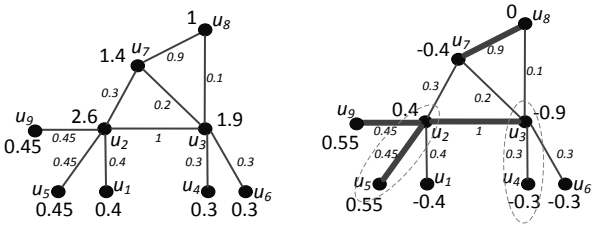
Given Lemma 1, a representative that preserves $[deg(\mathcal{G})]$ should contain $\mathbf{P}$ edges. Initially, ADR rounds $\mathbf{P}$ to the closest integer $\lfloor\mathbf{P}\rceil$ and sorts the edges on descending order of their probabilities. Consequently, it iterates through the sorted list, and samples each edge e with probability p_e, until it has included $\lfloor\mathbf{P}\rceil$ edges. Algorithm 2 illustrates the pseudocode of ADR, where lines 1-7 correspond to Phase 1.

Phase 2 starts with E_0. At each iteration i, let the current set of edges be E_i. For each node $u \in V$, ADR randomly picks two edges $e_1 = (u, v) \in E_i$ and $e_2 = (x, y) \in E \setminus E_i$ (line 10-11) and computes $d_1 \leftarrow |dis_u - 1| + |dis_v - 1| - (|dis_u| + |dis_v|)$ and $d_2 \leftarrow |dis_x + 1| + |dis_y + 1| - (|dis_x| + |dis_y|)$. Specifically, d_1 is the difference of the absolute discrepancies of u and v, caused by the removal of e_1. Accordingly, d_2 is the difference of the absolute discrepancies of x and y caused by the addition of edge e_2. ADR replaces e_1 with e_2 if $d_1 + d_2 < 0$, i.e., the swapping of edges decreases the overall discrepancy. Since the total number of edges remains $\lfloor\mathbf{P}\rceil$, the expected average degree of $\mathcal{G}$ is preserved throughout the process. The procedure terminates after a user-defined number of iterations *steps*. The value of *steps* depends on the desired trade-off between quality and efficiency. The running time of ADR is $O(|E|\log|V| + steps)$.

Algorithm 2 Average Degree Rewiring (ADR)

Input: uncertain graph $\mathcal{G} = (V, E, p)$, steps
Output: approximate representative $G^* = (V, E^*)$
1: $E_0 \leftarrow \emptyset$, $i \leftarrow 0$
2: $\mathbf{P} \leftarrow \sum_{e\in E} p_e$
3: sort E in non increasing order of their probabilities
4: **while** $|E_0| < \lfloor\mathbf{P}\rceil$ **do**
5: $e \leftarrow E.next()$; $r \leftarrow$ random number $\in [0, 1]$
6: **if** $r \le p_e$ **then**
7: $E_0 \leftarrow E_0 \cup e$
8: **for** $i = 1$..steps **do**
9: **for each** $u \in V$ **do**
10: pick a random edge $e_1 = (u, v)$ from E_i
11: pick a random edge $e_2 = (x, y)$ from $E \setminus E_i$
12: $d_1 \leftarrow |dis_u - 1| + |dis_v - 1| - (|dis_u| + |dis_v|)$
13: $d_2 \leftarrow |dis_x + 1| + |dis_y + 1| - (|dis_x| + |dis_y|)$
14: **if** $d_1 + d_2 < 0$ **then**
15: $E_{i+1} \leftarrow (E_i - \{e_1\}) \cup \{e_2\}$
16: $E^* \leftarrow E_i$

We illustrate the application of ADR on the uncertain graph of Figure 3(a), where edge probabilities are denoted with italics, and the expected degree is shown next to each vertex. Initially, ADR computes $\mathbf{P} = 4.4$ and rounds it to the closest integer $\lfloor\mathbf{P}\rceil = 4$. Then, it samples 4 probable edges of the graph and forms the set $E_0 = \{(u_2, u_3), (u_2, u_5), (u_2, u_9), (u_7, u_8)\}$. Figure 3(b) depicts the edges of E_0 with bold lines, and shows the resulting node discrepancies. The value of the total discrepancy at this stage is $\Delta = 3.8$. Next, ADR starts the second phase. Assume that at iteration 0 the algorithm randomly considers the replacement of $e_1 = (u_2, u_5) \in E_0$ with $e_2 = (u_3, u_4) \in E \setminus E_0$. Since $d_1 = 0.6 + 0.45 - (0.40 + 0.55) = 0.1$, $d_2 = 0.1 + 0.7 - (0.9 + 0.3) = -0.4$ and $d_1 + d_2 = -0.3 < 0$, the edges are swapped. Intuitively, the swapping reduces the overall discrepancy by $|d_1 + d_2|$. The discrepancy of the new instance $E_1 = \{(u_2, u_3), (u_2, u_9), (u_3, u_4), (u_7, u_8)\}$ is $\Delta' = \Delta - |d_1 + d_2| = 3.5$.



(a) initial graph (b) after Phase 1

Figure 3: ADR example

5. APPROXIMATE B-MATCHING (ABM)

This section presents ABM, which stands for Approximate B-Matching. We first discuss the motivation behind ABM, and then provide the algorithmic framework.

5.1 Motivation

Recall from Section 2.3 that the maximum b-matching problem takes as input a deterministic graph $G = (V, E)$

and a capacity constraint $b(u)$ for every $u \in V$. The output is a set of edges $E_m \subseteq E$, such that each vertex u of V is incident to at most b_u edges of E_m.

We next investigate the relationship of our problem to b-matching, starting with the special case where all the expected degrees of $\mathcal{G}$ are integers, i.e., $[deg_u] \in \mathbb{Z}, \forall u \in V$. As we shall prove shortly, the *optimal instance* (i.e., the one that minimizes the overall discrepancy Δ) is given by a maximum b-matching computed on the uncertain graph $\mathcal{G}$ with capacity constraints $[deg_u]$ for each vertex $u \in V$.

LEMMA 2. *Assume that $[deg_u] \in \mathbb{Z}, \forall u \in V$. Then, there is at least one optimal instance G^* for which $deg_u(G^*) \leq [deg_u]$, for all $u \in V$.*

PROOF. Assume an optimal solution $G^* = (V, E^*)$ that contains *illegal* vertices, i.e., vertices $u \in V$ with $deg_u(G^*) > [deg_u]$. E^* cannot contain an edge (u, v) between two illegal vertices u and v, otherwise G^* is not optimal (i.e., the exclusion of edge (u, v) would decrease the overall discrepancy Δ). Thus, an illegal vertex u can only be adjacent to *legal* vertices, i.e., vertices $x \in V$ for which $deg_x(G^*) \leq [deg_x]$. Assume an edge $e = (u, x) \in E^*$, where u is illegal and x is legal. We first prove that if we remove edge e from E^*, then the remaining graph $G' = (V, E^* - \{e\})$ is also an optimal instance.

Specifically, $dis_u(G^*) > 0$ whereas $dis_x(G^*) \leq 0$, and $dis_u(G') = dis_u(G^*) - 1 \geq 0$ whereas $dis_x(G') = dis_x(G^*) - 1 < 0$. The overall discrepancy of G^* is:
$\Delta(G^*) = |dis_u(G^*)| + |dis_x(G^*)| + \sum_{v \neq u, x \in V} |dis_v(G^*)| = dis_u(G^*) - dis_x(G^*) + \sum_{v \neq u, x \in V} |dis_v(G^*)|$.

Similarly, the discrepancy of G' is:
$\Delta(G') = |dis_u(G')| + |dis_x(G')| + \sum_{v \neq u, x \in V} |dis_v(G')| = (dis_u(G^*) - 1) + (1 - dis_x(G^*)) + \sum_{v \neq u, x \in V} |dis_v(G')|$.

Since graphs G^* and G' only differ by the edge (u, x), $\sum_{v \neq u, x \in V} |dis_v(G^*)| = \sum_{v \neq u, x \in V} |dis_v(G')|$, and thus $\Delta(G^*) = \Delta(G')$. By applying the above argument to all the illegal vertices of G^*, we construct an optimal instance that contains only legal vertices. □

THEOREM 1. *Let $\mathcal{G} = (V, E, p)$ be an uncertain graph where $[deg_u] \in \mathbb{Z}$, $\forall u \in V$. An optimal solution of the* REPRESENTATIVE INSTANCE *problem on input $\mathcal{G}$ is given by solving a maximum b-matching on graph $\mathcal{G}$ using $[deg_u]$ as capacity constraint of vertex u.*

PROOF. Using the previous lemma, there is always an optimal solution $G^* = (V, E^*)$ that ensures that $deg_u(G^*) \leq [deg_u]$, for all $u \in V$. Thus, we can remove the absolute values from the definition of Δ:

$$\Delta(G^*) = \sum_{u \in V} \left| deg_u(G^*) - [deg_u] \right| =$$

$$= \sum_{u \in V} \left([deg_u] - deg_u(G^*) \right) = \sum_{u \in V} [deg_u] - \sum_{u \in V} deg_u(G^*)$$

Since the expected degrees $[deg_u]$ are fixed, this is equivalent to maximizing $\sum_{u \in V} deg_u(G^*)$, which in turn leads to the maximization of $|E^*|$. Therefore, G^* is a maximum b-matching on $\mathcal{G}$ with capacity constraints $[deg_u]$. □

According to the above theorem, a maximum b-matching on graph $\mathcal{G}$ leads to the optimal solution if *all* expected degrees are integers. The fastest algorithm for this computation [25] has time complexity $O(|E|^{3/2})$.

5.2 Algorithm

Since actual uncertain graphs have real valued expected degrees, the b-matching technique of the previous section cannot be applied directly. Moreover, the exact solution to the real-valued problem is expected to be at least as hard as in the case of integers, which is already expensive ($O(|E|^{3/2})$) for large graphs. Thus, we propose a novel approximation technique, based on *matching*.

The main idea is to extract an approximate instance in two phases. In Phase 1, ABM rounds the expected vertex degrees to the closest integers, and computes a maximal b-matching using the rounded values as capacity constraints. In the second phase, ABM partitions the vertices according to their discrepancy. Then, it extracts additional edges that improve the total discrepancy Δ, by performing a bipartite matching. We use approximation techniques for the two phases (i.e., b-matching and bipartite matching) for efficiency reasons.

Algorithm 3 Approximate b-Matching (ABM)

Input: uncertain graph $\mathcal{G} = (V, E, p)$
Output: approximate representative $G^* = (V, E^*)$.
1: calculate the expected degree $[deg_i]$ for all vertices in V.
2: $E_m \leftarrow 0, deg_u \leftarrow 0$
// Phase 1
3: let $b_i = \mathsf{round}([deg_i])$ to the closest integer
4: **for each** $e = (u, v) \in E$ **do**
5: **if** $deg_u < b_u$ AND $deg_v < b_v$ **then**
6: $E_m \leftarrow E_m \cup \{e\}$
7: $deg_u \leftarrow deg_u + 1; deg_v \leftarrow deg_v + 1$
// Phase 2
8: $A \leftarrow \emptyset, B \leftarrow \emptyset, C \leftarrow \emptyset$
9: **for each** $u \in V$ **do**
10: let $dis_u = deg_u - [deg_u]$
11: **if** $dis_u \leq -0.5$ **then**
12: $A \leftarrow A \cup \{u\}$
13: **else if** $-0.5 < dis_u < 0$ **then**
14: $B \leftarrow B \cup \{u\}$
15: **else**
16: $C \leftarrow C \cup \{u\}$
17: $E' \leftarrow E \setminus E_m$
18: **for each** edge $e = (u, v) \in E'$ **do**
19: weight$= |dis_u| + 2|dis_v| - |1 + dis_u| - 1$
20: **if** $(u \in A)$ AND $(v \in B)$ AND (weight> 0) **then**
21: $w(e) \leftarrow$ weight
22: **else**
23: discard e
24: Let $G' = ((A \cup B), E', W)$ where $W : w(e) \rightarrow \mathbb{R}$
25: $E_{BP} = \mathsf{bipartite}(G')$
26: $E^* = E_m \cup E_{BP}$

Algorithm 3 contains the pseudocode of ABM. Phase 1 (lines 3-7) corresponds to a greedy approximate maximum b-matching [14] that considers all edges in random order. For each edge $e = (u, v)$, if the capacity constraints of both vertices u and v are not violated, then e is inserted into the result set E_m, and the degrees of u and v are incremented. After all edges have been considered, E_m contains a maximal b-matching of $\mathcal{G}$, whose cardinality is at least half of that of the maximum [14]. Figure 4 applies ABM on the

uncertain graph of Figure 3(a). Figure 4(a) shows the vertex degrees after rounding. ABM considers in turn edges $(u_2, u_3), (u_7, u_8)$, which are added to E_m. After that, no other edge can be included in E_m because it would cause a capacity violation. Figure 4(b) includes the node discrepancies after the termination of Phase 1, with respect to their original (i.e., before rounding) degree.

Based on their discrepancies, lines 8-16 partition the vertices into three groups A, B and C. A contains vertices with discrepancy $dis_u \leq -0.5$, B the vertices for which $-0.5 < dis_u < 0$, and C the rest, i.e., vertices with $dis_u \geq 0$. The partitioning is complete (i.e., $A \cup B \cup C = V$) and there is no overlap (i.e., $A \cap B \cap C = \emptyset$). In our running example the groups are $A = \{u_2, u_3\}$, $B = \{u_1, u_4, u_5, u_6, u_7, u_9\}$ and $C = \{u_8\}$.

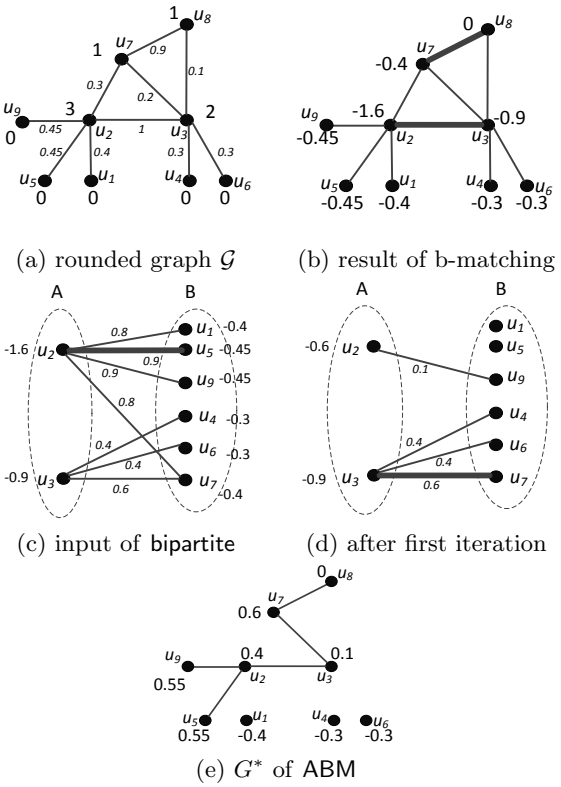


Figure 4: ABM example

Intuitively, A contains vertices, whose absolute discrepancy will decrease by the addition of an edge. B contains vertices whose absolute discrepancy will increase (after the addition of an edge) by less than 1. C contains vertices that have already reached or exceeded their expected degree[1]; thus, a new adjacent edge will increase their discrepancy by 1. Edges between vertices of A (e.g., (u_2, u_3)) have been added to the result set E_m during Phase 1. The following lemmas discuss the potential for including additional edges, depending on the group of their incident vertices.

LEMMA 3. *Let an edge (u, v) where $u, v \in B$. Including edge (u, v) in the result set cannot improve the overall discrepancy Δ.*

[1]Such vertices are possible because b-matching is performed on the rounded degrees.

PROOF. Since both vertices belong to the set B, it holds that $-0.5 < dis_i < 0$ for $i = \{u, v\}$. Thus, their total discrepancy is $d_1 = |dis_u| + |dis_v| < 1$. The addition of edge (u, v) will change the discrepancies to $dis'_u = dis_u + 1 > 0.5$, and $dis'_v > 0.5$. Thus, the total discrepancy becomes $d_2 = |dis'_u| + |dis'_v| > 1$. Since $d_1 < d_2$, edge (u, v) increases Δ. □

LEMMA 4. *Let an edge (u, v) where $u \in C$. Including edge (u, v) in the result cannot improve the overall discrepancy Δ.*

PROOF. Since u has exceeded its expected degree, adding edge (u, v) will increase dis_u by exactly 1. On the other hand, the absolute discrepancy of v can increase or decrease by at most 1. Thus, Δ can only increase (if $v \in B$ or $v \in C$) or remain the same (if $v \in A$). □

The above lemmas state that the inclusion in the result of an edge that connects i) two vertices in B or ii) a vertex in C with any vertex, cannot improve Δ. Therefore, after Phase 1, the only possible additions are edges that connect vertices in A with vertices in B. Let such an edge $e = (a, b)$ with $a \in A$ and $b \in B$: if the addition of e in the result set decreases the absolute discrepancy of a more than it increases the absolute discrepancy of b, then it improves Δ. The following lemma, quantifies this improvement.

LEMMA 5. *Let $e = (a, b)$ where $a \in A$ and $b \in B$. Including edge e in the result set changes the overall discrepancy Δ by $g_e = |dis_a| + 2|dis_b| - |dis_a + 1| - 1$.*

PROOF. The total discrepancy of vertices a and b *before* (a, b) is, $d_1 = |dis_a| + |dis_b|$. After adding the edge, it becomes $d_2 = |dis_a + 1| + (1 - |dis_b|)$. We define the *gain* g_e of e as the difference between the two discrepancies, i.e., $g_e = d_1 - d_2 = (|dis_a| + |dis_b|) - \big(|dis_a + 1| + (1 - |dis_b|)\big) = |dis_a| + 2|dis_b| - |dis_a + 1| - 1$. If g_e is positive, edge e decreases Δ; otherwise it increases it. If $dis_a \leq -1$, then $|dis_a + 1| = |dis_a| - 1$, and the gain becomes $g_e = 2|dis_b|$, i.e., g_e depends only on $|dis_b|$. □

An edge e can improve the overall discrepancy, only if $g_e > 0$. Lines 18-23 of Algorithm 3 consider each edge e in $E' = E - E_m$ connecting vertices in A and B. If $g_e > 0$, e is inserted in a graph $G' = (A \cup B, E', W)$ where $W : E' \to \mathbb{R}$, with $w(e) = g_e$. Figure 4(c) shows the graph G', including the vertex discrepancies and edge weights, which correspond to their gains. The next question is how to efficiently select the subset of edges in E' that minimizes Δ. Towards this, subroutine bipartite(G') (line 25) performs an approximate maximum weight bipartite matching on graph G' with a twist: a vertex of A may be matched with multiple vertices of B, if it has high absolute discrepancy.

Algorithm 4 illustrates bipartite. Initially, the edges of E' are sorted on non-increasing order of their weights/gains. Let Q be a priority queue that contains the sorted list. At each iteration, the head $e = (a, b)$ of Q is inserted into the result set. The inclusion of (a, b) changes the discrepancies of a and b. Specifically, the new discrepancy of b becomes positive; thus, according to Lemma 4, edges adjacent to b cannot reduce Δ, and are removed from Q. Then, bipartite updates the discrepancy of a; let dis_a be the new value. If $dis_a \leq -1$, from Lemma 5, the weights of the other edges (a, x) incident to a do not change since they depend only on $|dis_x|$. If $-1 < dis_a < -0.5$, bipartite updates the weights of

these edges using the *gain* definition of Lemma 5 (line 10). If a weight becomes negative, the edge is removed. Finally, if $dis_a > -0.5$, vertex a cannot further improve Δ as it does not belong to group A anymore; thus, it is discarded and all adjacent edges (a, x) are expunged from Q. The function terminates when Q becomes empty.

Continuing the example of Figure 4(c), bipartite first picks the heaviest edge (u_2, u_5) and adds it to the result. Then, it updates the discrepancy of u_2 to $dis_{u2} = -1.6 + 1 = -0.6$; since $-1 < dis_{u2} < -0.5$, the weights of edges adjacent to u_2, (i.e., $(u_2, u_1), (u_2, u_7)$ and (u_2, u_9)) must be updated as well. Edges (u_2, u_1) and (u_2, u_7) yield a negative weight, and are discarded. Figure 4(d) shows the bipartite graph after the first iteration. The next edge (u_3, u_7) in Q is included in the result, updating the discrepancy of vertex u_3 to $dis_{u3} = -0.9 + 1 = 0.1$. Since $dis_{u3} > 0$, all edges adjacent to u_3 are discarded. Finally, bipartite extracts the last edge (u_2, u_9) of Q, adds it to the result and terminates.

Figure 4(e) shows the final output of ABM, which combines the edges added during the two phases. The discrepancy of the extracted graph is 3.2. The total cost of ABM includes the linear-time processing of edges in Phase 1, and the sorting and queuing operations of Phase 2 on E'. Each edge of E' can be processed at most $|B|$ times. Therefore, the complexity of ABM is $O(|E| + |B||E'| \log |E'|)$, where $|E'| \leq |E|$.

Algorithm 4 bipartite

Input: bipartite graph $\mathcal{G} = (A \cup B, E', W)$
Output: the set of edges $E_{BP} \subseteq E'$ with high gain for Δ

```
1: E_BP ← ∅
2: sort the edges e ∈ E' on non-increasing order of their
   weights w(e) and add them in a priority queue Q.
3: while Q ≠ ∅ do
4:     e = (a, b) ← Q.next()
5:     E_BP ← E_BP ∪ {e}
6:     discard all edges in Q incident to b
7:     dis_a ← dis_a + 1
8:     if −1 < dis_a < −0.5 then
9:         for each edge e' = (a, x) ∈ E' do
10:            w(e') ← |dis_a| + 2|dis_x| − |dis_a + 1| − 1
11:            if w(e') > 0 then
12:                update order of e' in Q
13:            else
14:                discard e'
15:    else if dis_a > −0.5 then
16:        for each edge e' = (a, x) ∈ E' do
17:            discard e'
```

6. EXPERIMENTS

In this section, we compare ADR (*Average Degree Rewiring*) and ABM (*Approximate B-Matching*) against the two benchmark approaches MP (*Most Probable*) and GP (*Greedy Probability*). Our goal is to show that in addition to minimizing the node discrepancy, the representatives preserve other important graph properties. To this end, we measure the accuracy of the methods on the following metrics:

- *Vertex degree* corresponds to the percentage of vertices having a certain degree value. It is the property most closely related to the objective function of our algorithms, and its significance to the graph structure has been well established [22].
- *Clustering coefficient* is a measure of how close neighbors of the average k-degree vertex are to forming a clique. Specifically, it is the ratio of the average number of edges between the neighbors of k-degree vertices to the maximum number of such links. It is an important metric for search strategies [11] and social networks [19].
- *Shortest-path distance* is the percentage of pairs at a certain distance, over all pairs of reachable vertices. This metric is crucial for spatial queries [29], routing protocols, and in general, any task involving shortest path computations.
- *Betweenness centrality* is a measure of the node's importance in the graph: it corresponds to the ratio of shortest paths that pass through the node over all pairs of shortest paths. It has been used widely to assess link value of ecological graphs [12], and router utilization of communication networks [31].

Table 2: Characteristics of real datasets

dataset	*vertices*	*edges*	*edge probabilities: mean, SD, quartiles*
BioMine	1 008 201	13 485 878	0.27 ± 0.21, $\{0.12, 0.22, 0.36\}$
DBLP	684 911	4 569 982	0.14 ± 0.11, $\{0.09, 0.09, 0.18\}$
Flickr	78 322	10 171 509	0.09 ± 0.06, $\{0.06, 0.07, 0.09\}$

The *ground truth* (i.e., the expected value of the measure on the uncertain graph) can be computed exactly only for the *vertex degree*, by summing up the probabilities of the adjacent edges for each vertex. All the other metrics are approximated by Monte Carlo sampling. Specifically, we create a number of random instances of the input uncertain graph, and we approximate the *expected* value of each metric by the weighted average of the sampled graphs. We use 1000 samples since it has been shown [29, 16] that they usually suffice to achieve accuracy convergence. However, *shortest-path distance* and *betweenness centrality* are very expensive because they involve *all-pairs-shortest-path* computations. Their evaluation over 1000 samples of large graphs is prohibitive. To overcome this problem, given an uncertain graph, we use *forest fire* [20] to create a smaller subgraph that has similar properties and we perform the evaluation on that subgraph.

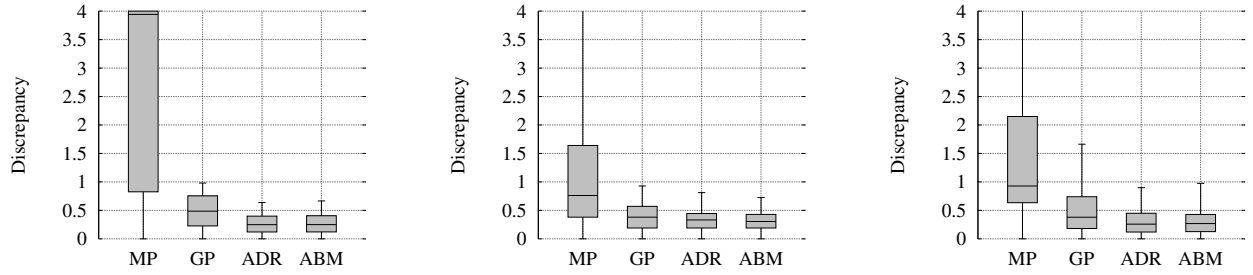
Section 6.1 compares our techniques on *vertex degree* and *clustering coefficient*, using real uncertain graphs. Section 6.2 applies *forest fire* to reduce the size of the real graphs, in order to evaluate *shortest-path distance*, and *betweenness centrality*. Finally, Section 6.3 considers all metrics on synthetic graphs.

6.1 Real graphs

Table 2 summarizes the main characteristics of the uncertain graphs from different real world scenarios, used in our experiments.

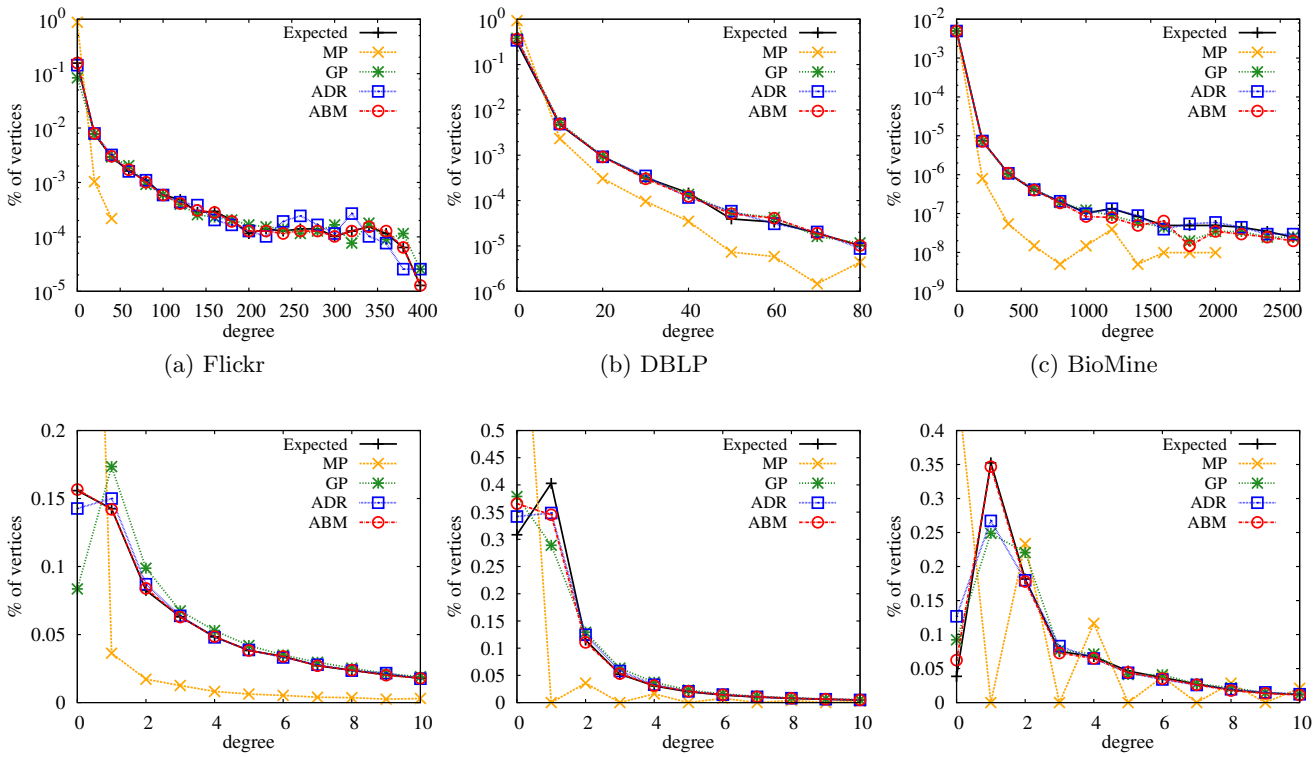
Flickr [29](`www.flickr.com`): a social network, where the probability of an edge between two users is computed assuming *homophily*, i.e., the principle that similar interests indicate social ties. Homophily is measured by the Jaccard coefficient of the interest groups of the two users.

DBLP [29, 16] (`www.informatik.uni-trier.de/~ley/db/`): a database of scientific publications and their authors. Two



(a) Flickr (b) DBLP (c) BioMine

Figure 5: Box plots of the distribution of discrepancy per node



(a) Flickr (b) DBLP (c) BioMine

(d) Flickr (e) DBLP (f) BioMine

Figure 6: Vertex degree (a)-(c) and zoom-in (d)-(f) for most probable degrees (real graphs)

authors are connected, if they have coauthored a publication. The probability of an edge is derived from an exponential function to the number of collaborations.

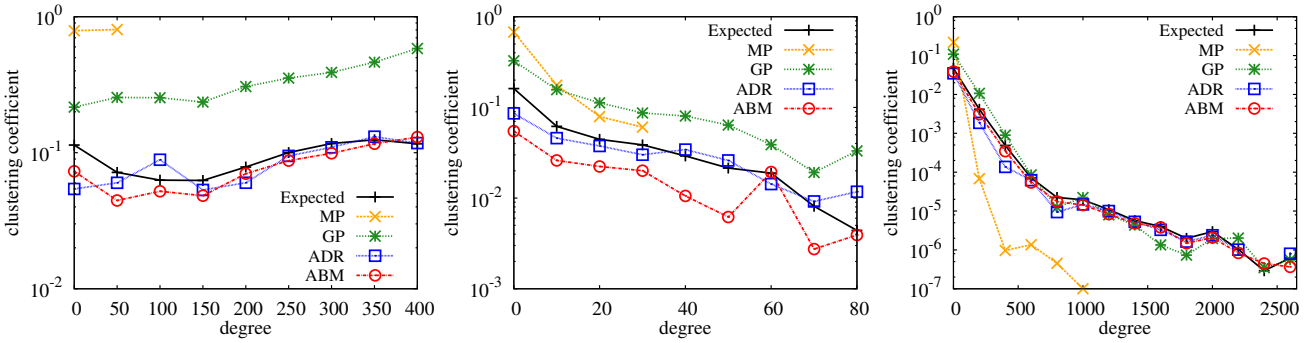
BioMine [30] (`biomine.org`): a snapshot of the database of the BioMine project containing biological interactions. The probability of any edge corresponds to the confidence that the interaction actually exists.

Figure 5 illustrates the boxplots for the absolute discrepancy distribution. For each method, the vertical line includes 96% of the vertices, the gray rectangle contains 50% of the vertices, and the horizontal line corresponds to the median discrepancy. For instance, in Flickr for the representative produced by GP, 96% of the vertices have absolute discrepancy less than 0.98, 50% of the vertices in the range [0.23, 0.76], and the median discrepancy is 0.48. As expected, the accuracy of MP is very low; the upper bound of the 96% range reaches discrepancy 129 for Flickr and is omitted from Figure 5. GP performs significantly better, but it is clearly outperformed by ADR and ABM, whose median discrepancy is below 0.4, in all datasets.

Table 3: Average discrepancy per vertex

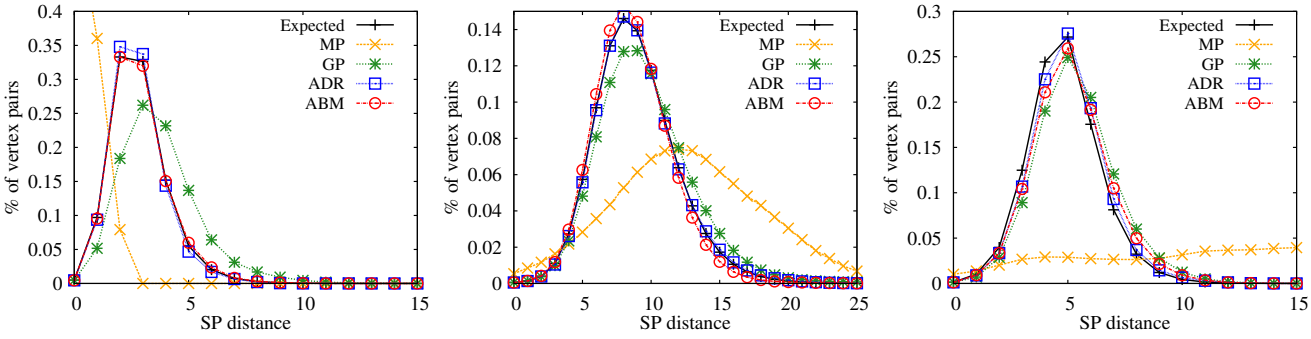
	Flickr	DBLP	BioMine
MP	22.108	1.599	4.260
GP	0.560	0.407	0.678
ADR	0.278	0.352	0.346
ABM	0.280	0.315	0.453

Table 3 shows the average discrepancy per vertex for all datasets. Note that for Flickr the average discrepancy of MP is much higher than the median. This occurs because, as we show in the next experiment, the highest vertex degree of the representative is much lower than the highest expected degree in the uncertain graph. The accuracy of GP is reason-



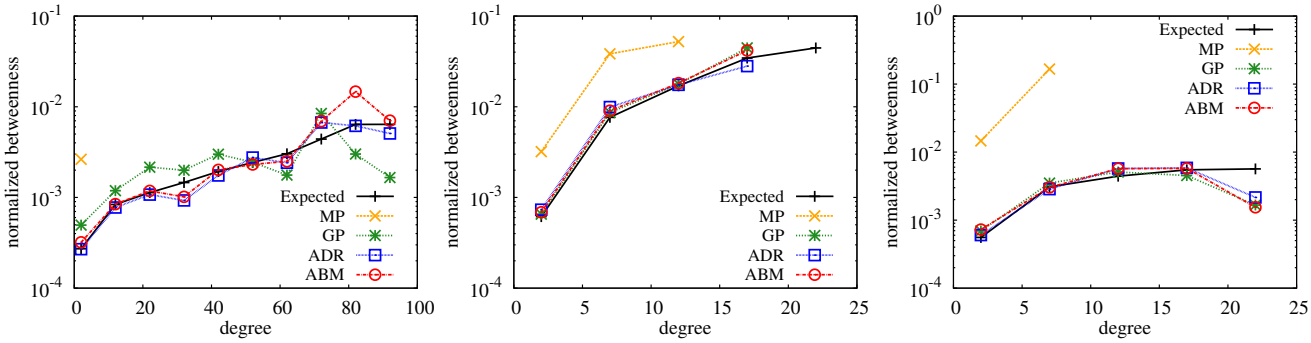
(a) Flickr (b) DBLP (c) BioMine

Figure 7: Clustering coefficient (real graphs)



(a) Flickr (b) DBLP (c) BioMine

Figure 8: Shortest path distance (real reduced graphs)



(a) Flickr (b) DBLP (c) BioMine

Figure 9: Betweeness centrality (real reduced graphs)

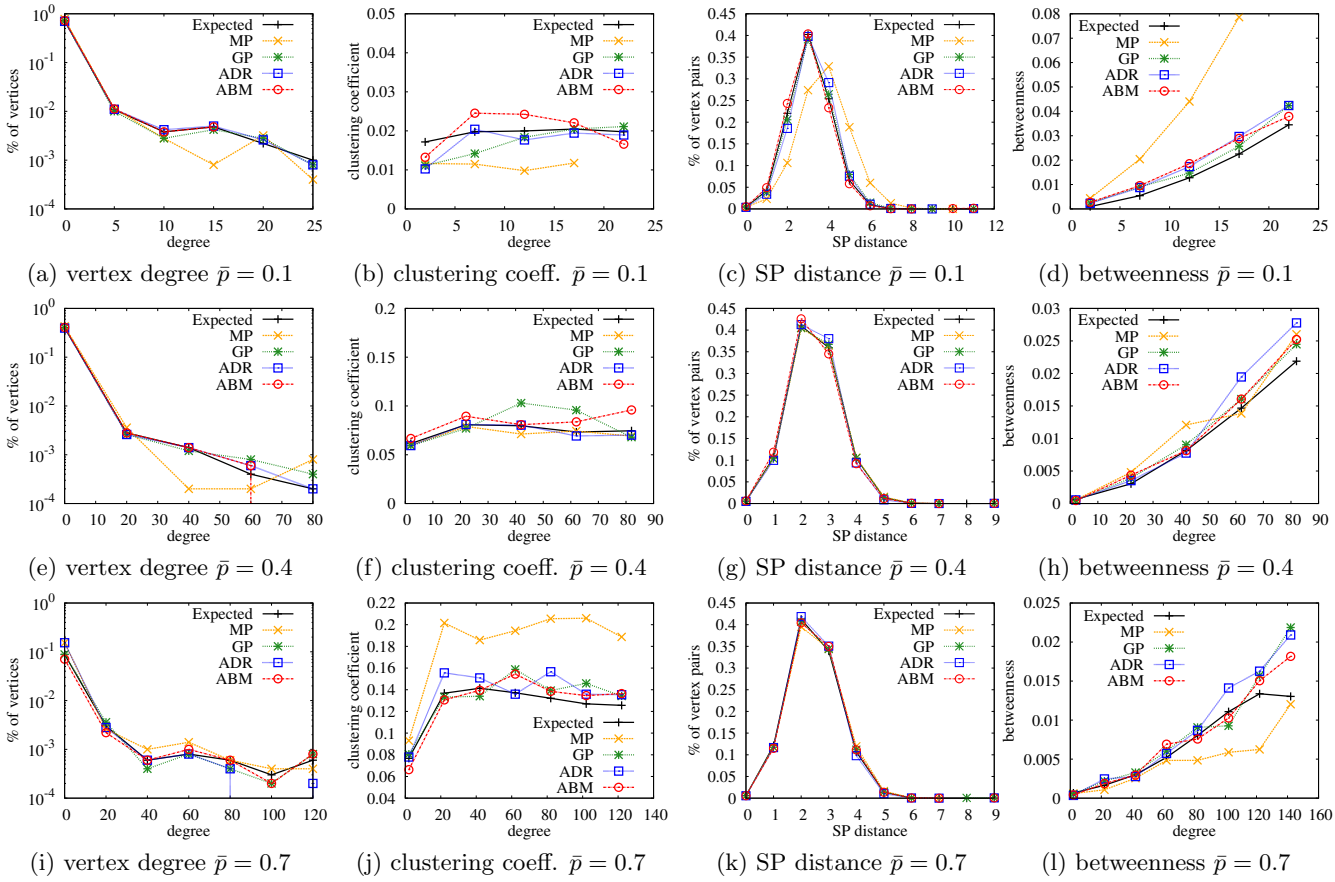
able, but below that of ADR and ABM for all datasets. ADR and ABM have similar performance for Flickr. For DBLP (BioMine), ABM (ADR) yields better results.

Figure 6 shows the degree distribution, i.e., the percentage of vertices in the representative instance versus the vertex degree. The second row focuses on degrees 0 to 10, which are the most probable. ADR and ABM are very close to the expected distributions in all datasets. Naturally, MP is the worst method; for instance, in Flickr the highest degree of any vertex of its representative is below 50, although there are vertices whose expected degree is up to 400. The inferior performance of GP with respect to ADR and ABM is evident in the zoomed diagrams.

Figure 7 illustrates the clustering coefficient versus the vertex degree. In general, the ranking of the algorithms in terms of accuracy, is the same as that for the node discrepancy. The performance of MP is clearly unacceptable. GP performs well for BioMine but it yields large error for Flickr and DBLP. ADR and ABM achieve the best clustering coefficient approximations, with ADR outperforming ABM for DBLP, which is consistent with the average discrepancy of the algorithms in Table 3.

6.2 Reduced real graphs

In order to evaluate *shortest path (SP) distance* and *betweenness centrality*, we produce subgraphs of the original dataset using *forest fire* [20]. The number of vertices and edges in the resulting subgraphs are: i) Flickr, 5000 and 655275, ii) BioMine, 5000 and 69367, and iii) DBLP, 20000 and 79619. Note that the method used to create reduced



(a) vertex degree $\bar{p} = 0.1$ (b) clustering coeff. $\bar{p} = 0.1$ (c) SP distance $\bar{p} = 0.1$ (d) betweenness $\bar{p} = 0.1$

(e) vertex degree $\bar{p} = 0.4$ (f) clustering coeff. $\bar{p} = 0.4$ (g) SP distance $\bar{p} = 0.4$ (h) betweenness $\bar{p} = 0.4$

(i) vertex degree $\bar{p} = 0.7$ (j) clustering coeff. $\bar{p} = 0.7$ (k) SP distance $\bar{p} = 0.7$ (l) betweenness $\bar{p} = 0.7$

Figure 10: All metrics for different average edge probability (synthetic graphs)

graphs is orthogonal to our work; we expect the proposed techniques to have similar performance with other size reduction methods.

Figure 8 illustrates the shortest path distance distribution, i.e., the percentage of vertices versus the SP distance. Figure 9 shows the betweenness centrality versus the vertex degree. In general, the results are consistent with those of the previous subsection. Specifically, the representatives of ADR and ABM capture very well both metrics in all datasets. Similar to Figure 7, GP performs well for BioMine and poorly for Flickr. However, its accuracy for *SP distance* and *betweeness centrality* in DBLP is rather good, as opposed to the *clustering coefficient* for the same dataset. This indicates that a representative may be suitable for some tasks but not others.

6.3 Synthetic graphs

To create synthetic uncertain graphs, we first generate a degree sequence $d_1, \ldots, d_{|V|}$ by sampling from a power-law distribution. Then, we use the Chung-Lu model [8] to build a graph with the specific degree sequence: for each pair of vertices u and v, we draw the corresponding edge (u, v) with probability proportional to $d_u \cdot d_v$. Finally, for every edge, we assign a probability value from a normalized Zipf distribution because it has been shown that it captures well real-world properties [7].

We experimented with two parameters: i) the density δ defined as the ratio between the number of edges and the number of vertices, and ii) the average edge probability $\bar{p}$. Due to lack of space, we only show the results after fixing density to $\delta = 5$ and varying the average edge probability. In all cases, the node cardinality is fixed to 10000.

Figure 10 measures the accuracy of all algorithms on all metrics for the synthetic data. Specifically, each row corresponds to an average probability value and each column to a metric. Similar to the previous experiments, ADR and ABM produce the best representative. The main difference is that for the synthetic data, GP also performs well in most cases. On the other hand, MP is acceptable only for $\bar{p} = 0.4$ because, if the edge probabilities are close to 0.5, MP is likely to include the expected number of edges in the representative.

Comments on the run-time. The running time of the algorithms is negligible compared to the cost of query processing. A single core of an Intel Xeon server at 2.83GHz CPU and 64GB RAM takes about one minute to generate a representative instance using ADR (the most expensive algorithm[2] on BioMine (the largest dataset with 13M edges). ABM takes around 45 seconds, while the other algorithms are even faster. On the other hand, just the generation of 1000 samples requires several minutes. Moreover, the evaluation of the ground truth on 1000 samples is 1000 times more expensive than the evaluation of the corresponding metric on a representative.

[2]In our implementation we set the *steps* of ADR equal to $10 \cdot \mathbf{P}$

7. CONCLUSION

In this paper, we introduce the problem of extracting representative instances of uncertain graphs. Expensive tasks can then be processed by applying deterministic algorithms on these instances. We propose methods that aim at capturing the expected degree of individual vertices because of its significance in the graph topology. An extensive experimental evaluation with real and synthetic data confirms that the representative instances indeed preserve well a number of important graph metrics.

We intend to investigate the complexity class of the problem as well as approximation bounds for the representatives, which are complicated issues beyond the scope of this work. We will also study additional algorithms that could possibly lead to better results. Finally, an interesting direction for future work concerns specialized algorithms that aim at a specific type of task, e.g., data mining, rather than general graph metrics or properties.

Acknowledgements

Panos Parchas and Dimitris Papadias were supported by grant HKUST 6177/13 from Hong Kong RGC.

8. REFERENCES

[1] S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. In *SIGMOD*, pages 34–48, 1987.

[2] E. Adar and C. Re. Managing uncertainty in social networks. *IEEE Data Eng. Bull.*, 30(2):15–22, 2007.

[3] C. C. Aggarwal and P. S. Yu. A survey of uncertain data algorithms and applications. *TKDE*, 21(5):609–623, 2009.

[4] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth. Predicting protein complex membership using probabilistic network reliability. *Genome Res.*, 14:1170–1175, 2004.

[5] J. Blitzstein and P. Diaconis. A sequential importance sampling algorithm for generating random graphs with prescribed degrees. *Internet Mathematics*, 6(4):489–522, 2011.

[6] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa. Injecting uncertainty in graphs for identity obfuscation. *PVLDB*, 5(11):1376–1387, 2012.

[7] L. Chen and C. Wang. Continuous subgraph pattern search over certain and uncertain graph streams. *TKDE*, 22(8):1093–1109, 2010.

[8] F. R. K. Chung and L. Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1):91–113, 2003.

[9] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875, 2004.

[10] P. Erdös and T. Gallai. Graphs with prescribed degrees of vertices (hungarian). *Mat. Lapok*, 11:264–274, 1960.

[11] P. Fraigniaud. Small worlds as navigable augmented networks: model, analysis, and validation. In *ESA*, pages 2–11, 2007.

[12] A. M. M. Gonzalez, B. Dalsgaard, and J. M. Olesen. Centrality measures and the importance of generalist species in pollination networks. *Ecological Complexity*, 7(1):36–43, 2010.

[13] S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph (i). *J. Soc. Indust. Appl. Math.*, 10(3):496–506, 1962.

[14] S. Hougardy. Linear time approximation algorithms for degree constrained subgraph problems. In W. Cook, L. Lovasz, and J. Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 185–200. Springer, 2009.

[15] R. Jin, L. Liu, and C. C. Aggarwal. Discovering highly reliable subgraphs in uncertain graphs. In *KDD*, pages 992–1000, 2011.

[16] R. Jin, L. Liu, B. Ding, and H. Wang. Distance-constraint reachability computation in uncertain graphs. *PVLDB*, 4(9):551–562, 2011.

[17] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[18] J. Kleinberg. Complex networks and decentralized search algorithms. In *Int. Congr. of Mathematicians (ICM)*, 2006.

[19] G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. *Science*, 311(5757):88–90, 2006.

[20] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *KDD*, pages 631–636, 2006.

[21] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.

[22] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat. Systematic topology analysis and generation using degree correlations. *SIGCOMM Comput. Commun. Rev.*, 36(4):135–146, 2006.

[23] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, k. c. claffy, and A. Vahdat. The internet as-level topology: Three data sources and one definitive metric. *SIGCOMM Comput. Commun. Rev.*, 36(1):17–26, Jan. 2006.

[24] J. Mestre. Greedy in approximation algorithms. In *ESA*, pages 528–539, 2006.

[25] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *FOCS*, pages 17–27, 1980.

[26] D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Trans. on Information Theory*, 47(3):1212–1215, 2001.

[27] M. Mihail and N. K. Vishnoi. On generating graphs with prescribed vertex degrees for complex network modelling. *ARACNE*, pages 1–11, 2002.

[28] M. E. Newman. Spread of epidemic disease on networks. *Phys. Rev. E*, 66(1), 2002.

[29] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. k-Nearest Neighbors in uncertain graphs. *PVLDB*, 3(1):997–1008, 2010.

[30] P. Sevon, L. Eronen, P. Hintsanen, K. Kulovesi, and H. Toivonen. Link discovery in graphs derived from biological databases. In *DILS*, pages 35–49, 2006.

[31] A. Tizghadam and A. Leon-Garcia. Betweenness centrality and resistance distance in communication networks. *IEEE Network*, 24(6):10–16, 2010.

[32] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. on Computing*, 8(3):410–421, 1979.

[33] D. J. Watts. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[34] Y. Yuan, L. Chen, and G. Wang. Efficiently answering probability threshold-based shortest path queries over uncertain graphs. In *DASFAA*, pages 155–170, 2010.

[35] Y. Yuan, G. Wang, L. Chen, and H. Wang. Efficient subgraph similarity search on large probabilistic graph databases. *PVLDB*, 5(9):800–811, 2012.

[36] Y. Yuan, G. Wang, H. Wang, and L. Chen. Efficient subgraph search over large uncertain graphs. *PVLDB*, 4(11):876–886, 2011.

[37] Z. Zou, J. Li, H. Gao, and S. Zhang. Finding top-k maximal cliques in an uncertain graph. In *ICDE*, pages 649–652, 2010.

[38] Z. Zou, J. Li, H. Gao, and S. Zhang. Mining frequent subgraph patterns from uncertain graph data. *TKDE*, 22(9):1203–1218, 2010.