Breaking Cycles in Noisy Hierarchies*

Jiankai Sun
The Ohio State University
sun.1306@osu.edu

Deepak Ajwani
Nokia Bell Labs, Ireland
deepak.ajwani@nokia-bell-labs.com

Patrick K. Nicholson
Nokia Bell Labs, Ireland
pat.nicholson@nokia-bell-labs.com

Alessandra Sala
Nokia Bell Labs, Ireland
alessandra.sala@nokia-bell-labs.com

Srinivasan Parthasarathy
The Ohio State University
srini@cse.ohio-state.edu

ABSTRACT

Taxonomy graphs that capture hyponymy or meronymy relationships through directed edges are expected to be acyclic. However, in practice, they may have thousands of cycles, as they are often created in a crowd-sourced way. Since these cycles represent logical fallacies, they need to be removed for many web applications. In this paper, we address the problem of breaking cycles while preserving the logical structure (hierarchy) of a directed graph as much as possible. Existing approaches for this problem either need manual intervention or use heuristics that can critically alter the taxonomy structure. In contrast, our approach infers graph hierarchy using a range of features, including a Bayesian skill rating system and a social agony metric. We also devise several strategies to leverage the inferred hierarchy for removing a small subset of edges to make the graph acyclic. Extensive experiments demonstrate the effectiveness of our approach.

CCS CONCEPTS

•**Theory of computation** → *Network flows;* •**Computing methodologies** → *Ontology engineering;* •**Information systems** → Data cleaning;

KEYWORDS

Directed Acyclic Graph, Graph Hierarchy, TrueSkill, Social Agony, Cycle Edges

ACM Reference format:
Jiankai Sun, Deepak Ajwani, Patrick K. Nicholson, Alessandra Sala, and Srinivasan Parthasarathy. 2017. Breaking Cycles in Noisy Hierarchies. In *Proceedings of WebSci'17, June 25-28, 2017, Troy, NY, USA., ,* 10 pages.
DOI: http://dx.doi.org/10.1145/3091478.3091495

1 INTRODUCTION

A large number of applications in information science, in fields such as AI, semantic web, biomedical informatics, library science and information architecture, rely on ontological knowledge such as taxonomies and meronomies. Taxonomies capture generalization/specification of semantic concepts and categories and meronomies capture "has a" and "is a part of" relationship. Thus, the taxonomy and meronomy graphs (as well as graphs representing many other ontological relationships) should ideally be acyclic as cycles represent logical contradictions.

However, many large ontological knowledge bases are created either in (i) crowd-sourced way (e.g., Wikipedia categories [39]) or (ii) using automated text analytics tools (e.g, Yago [13, 31]). The creation process often results in inconsistencies and errors. As a result, the directed graphs capturing these ontological relations, can have a large number of cycles. For instance, the popular skos:broader category[1] is not guaranteed to be transitive and irreflexive and, as a result, it has various cycles that can "represent a potential problem for many web applications" [2]. In fact, there exists a body of work [5, 9, 20, 24, 26, 29, 33, 34, 38, 42] that has identified and recognized the presence of cycles in the hierarchical relations as one of the main problems for many web applications dealing with ontologies, such as United Medical Language System (UMLS) Metathesaurus graph and DBpedia taxonomy graph. For example, the unsupervised learning approach proposed by Fossati et al. [9], to automatically derive a taxonomy for a DBpedia entity from a prominent subset of the Wikipedia category graph, requires that the Wikipedia category graph is a directed acyclic graph (DAG), which can ensure a strict hierarchy. To support these web applications, there is a need for a principled technique to reduce a directed graph modeling a hierarchical relationship into a DAG, which only includes acyclic relationships.

In addition to removing logical contradictions, a DAG with strict hierarchy, can also benefit many applications to be computationally more efficient. This is because a DAG supports faster traversals (e.g., for computing descendants and transitive closure) compared to general directed graphs.

The problem of reducing graphs modeling hierarchical relations into DAGs is also relevant in many other domains. For instance, in synchronous dataflow (SDF) scheduling (an important problem in design automation for communication and digital signal processing systems [14, 15]), the existence of cycles in SDF graphs prevents or greatly restricts applications of many useful optimization techniques that are available for acyclic SDF graphs, because cyclic data dependences can cause deadlocks when scheduling tasks.

In this paper, we propose techniques to remove the cycles while preserving the logical structure (hierarchy) of a directed graph as

*This Work was conducted while the first author was doing internship at Nokia Bell Labs, Ireland.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WebSci'17, June 25-28, 2017, Troy, NY, USA.
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
978-1-4503-4896-6/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3091478.3091495

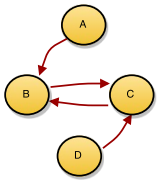
[1] https://www.w3.org/2009/08/skos-reference/skos.html
[2] https://www.w3.org/TR/skos-reference/#L2484

much as possible. Despite the proliferation of applications relying on ontologies capturing hierarchical relationships, there is hardly any solution for this problem that is principled, fast and fully automated. Existing approaches for this problem fall into the following categories:

- Simple DFS or BFS based heuristic (e.g., [9, 33, 34, 42]) to eliminate cycles (e.g., remove back edges in DFS)
- Theoretical solutions that model the problem in terms of variants of minimum-feedback arc set problem [18] or other NP-hard optimization problems
- Complex domain-specific algorithms (e.g. [24, 26]) that eliminate cycles based on many criteria, including redundancy and confidence of sources asserting the relations

While the first set of techniques are fast and automated, they are not principled – the edges removed heavily depend on the order of traversal, independent of other structural properties (For example in Figure 1, edge (C, B) will be deleted if DFS starts from node A, otherwise edge (B, C) will be the back edge). As a result, they often eliminate a large number of edges relevant for the taxonomy structure. The second set of techniques have a theoretical basis, but they are too slow in practice. Also, since minimum feedback arc set problem is APX-hard [17], one has to rely on approximation algorithms [8] with poor worst-case guarantees or heuristics (e.g., [2, 21, 30]). As such, it is not clear how well these techniques can preserve the logical structure of taxonomy while removing the cycles. The last set of techniques are (i) typically domain-specific, (ii) require significant manual intervention making it difficult to scale and (iii) depend on additional information (e.g., number of sources asserting each relation and confidence on sources) that is not easily available.

Figure 1: DFS-based Approach Toy Example



To overcome the above limitations, we propose to remove cycle edges via graph hierarchy. The graph hierarchy can be inferred by a Bayesian skill rating system (*TrueSkill*) [12] or Social agony (*SocialAgony*) [11], without any manual intervention. Several strategies are then devised to remove cycle edges based on their corresponding nodes' ranking scores in the hierarchy. To summarize, the key contributions of this article are:

- We address the problem of breaking cycles from directed graphs, while preserving the underlying hierarchy of the ontological relation as much as possible. To this end, we propose an ensemble of strategies, based on inferring the underlying graph hierarchy, to select edges for removal from the graph.
- To infer graph hierarchy, we (i) model it as a competition problem and leverage TrueSkill to solve it and (ii) model it as the problem of minimizing agony in social networks and leverage SocialAgony to solve it.
- We show extensive experimental results on real and synthetic data sets demonstrating the effectiveness of our approaches.

The rest of this article is organized as follows: In section 2, we describe related work. In section 3, we discuss ways to infer graph hierarchy and strategies to remove cycle edges via graph hierarchy. Section 4 reports on our empirical results. Finally, we conclude in Section 5.

2 RELATED WORK

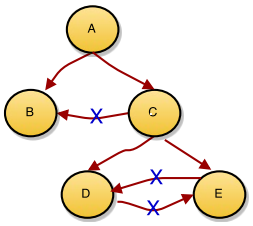
We categorize the related work as follows:

2.1 Simple Heuristics Based on BFS or DFS

Owing to their simplicity, speed and domain independence, simple heuristics based on Depth-First Search (DFS) (e.g., [33, 34, 42]) or Breadth-First Search (BFS) (e.g., [9]) have been used to remove cycles. Assuming edges in taxonomy graph point from specific to general concepts, DFS based approaches perform a DFS starting from the most specific concepts (zero in-degree) in the hierarchy and then remove the back edges. In contrast, a BFS based approach traverses the graph in a breadth-first manner (though still from the most specific concept l) and removes the edges (u, v) with $d(u) \geq d(v)$. Here, (u, v) is a directed edge from u to v, and $d(u)$ is the unweighted distance from l to u that is computed during the BFS traversal. Note that this can potentially remove many non-cycle edges as well and even from cycles, it may remove more edges than is strictly necessary. Take Figure 2 as an example: a BFS will start from a leaf node A, and will remove non-cycle edge (C, B) and both cycle edges (D, E) and (E, D).

The main problem with these heuristics is that there is no intuitive or empirical evidence that edges deleted this way are actually the appropriate edges to be removed. In our experiments, we found that these heuristics removed a large number of edges that were relevant for the taxonomy hierarchy, based on a manual insepection of the removed edges. The selection of edges to remove solely depends on the order in which the graph is traversed and is independent of other structural properties of the nodes.

Figure 2: BFS-based Approach Toy Example



2.2 Minimum Feedback Arc Set

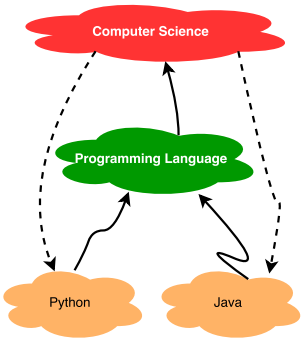
Theoretically, one can consider the problem of maximizing the size of the acyclic subgraph or minimizing the number of edges removed to make the graph acyclic. This is the popular minimum feedback arc set (MFAS) problem and it is on the original list of Richard M. Karp's 21 NP-complete problems [18]. In fact, the MFAS problem is APX-hard: Unless $P = NP$, the MFAS problem does not have

a polynomial-time approximation scheme (PTAS) [17]. Heuristic approaches are proposed to gradually build the feedback edge set by always removing the edge that breaks the most of the remaining simple cycles(e.g., [21]).

To avoid enumerating all simple cycles of a graph, there exists a greedy local heuristic method that only uses local information to make greedy choices [6, 7]. Score functions are defined based on local information (in- and out-degree of nodes). Such score functions are defined as: $score(i) = |d_i^{in} - d_i^{out}|$ or $score(i) = max(\frac{d_i^{in}}{d_i^{out}}, \frac{d_i^{out}}{d_i^{in}})$, where $score(i)$ is the score of a node i (a higher score means node i is more asymmetric regarding to its in- and out-degrees), and d_i^{in} (resp. d_i^{out}) are the in-degree (resp. out-degree) of node i. The node with the highest score is selected, then all of its in- or out-edges are removed, whichever edge set is of smaller cardinality.

MFAS heuristics (e.g., [6, 7, 21, 27, 35]) do not offer any guarantees on the number of edges removed and in the worst case, the gap between the heuristic solution and the optimal solution can be extremely large. Furthermore, even if an exact algorithm (e.g., [2]) can be used, there is no evidence to suggest that it preserves the logical hierarchy structure or that minimizing the edges to remove is the correct objective to optimize for cleaning the ontologies with hierarchical relations. Take Figure 3 for example, the edge (*Programming Language*,*Computer Science*) will be removed according to the above greedy local heuristic method. However, edges (*Computer Science*, Python) and (*Computer Science, Java*) should be removed to maintain the correctness of this graph's logical structure.

Figure 3: Minimum Feedback Arc Set Toy Example



2.3 Domain-specific Algorithms

Many other algorithms have been proposed for the problem of eliminating cycles for specific domains. However, these techniques rely on manual intervention and/or additional information that is often not available.

For UMLS Metathesaurus graph, solutions [24, 26] have been proposed for eliminating inappropriate edges causing circular hierarchical relations. The algorithm proposed by Bodenreider [26] is relatively complex and for complex cycles, it requires manual intervention [24] by domain experts, which is not scalable. The technique by Mougin and Bodenreider [24] uses a set of heuristics and rules to identify and eliminate all cycles from the UMLS graph. For example, criteria redundancy (i.e. the number of sources asserting each relation) and criteria confidence can be exploited to determine relations. However, this information about number and confidence of sources is not easily available for many ontological knowledge bases.

For synchronous data flow graphs, Bhattacharyya et al. and Hsu et al. [3, 14, 15] proposed the loose interdependence algorithm framework (LIAF) to decompose and break cycles. This framework relies on a very specific property of SDF graphs, namely, that for each strongly connected component (SCC)[3] of the SDF graph, there exists a set of edges with sufficient delays (an edge in an SDF graph is associated with a delay that can also be interpreted as the number of initial tokens) whose removal reduces the number of nodes in the SCC of the remaining graph. In fact, the algorithm recursively eliminates such edges to get rid of all cycles in the input SDF graphs (removed edges become inter-iteration edges). However, in our scenario of large ontological knowledge graphs, no such delay measure is known to exist and there is no corresponding notion of iterations, so these algorithms are not applicable.

Some approaches [20] transform the SCCs in the hierarchical relationships (broader/narrower, whole/part, generic/specific, instance of) into *related* relationship between the concepts. While this may be sufficient for some applications, many applications will end up simply ignoring the *related* relationship edges and the large number of corresponding direct and implied relations, leading to inaccurate results.

3 OUR APPROACH

In this section, we propose graph hierarchy based strategies to break cycles from a directed graph, while preserving the underlying hierarchy of the relations as much as possible. Consider a ranking function f that assigns a ranking score to each node in the graph. A higher ranking score implies that the corresponding node is higher up (e.g., more general) in the hierarchy. Given such a ranking, the edges which violate the hierarchy (i.e., edges from a higher/general group to a lower/specific group) are potential candidates for removal. Thus in our approach, there are two sub-tasks involved:

- Inferring graph hierarchy (or finding a ranking function)
- Proposing strategies to select violation edges as candidates for removal

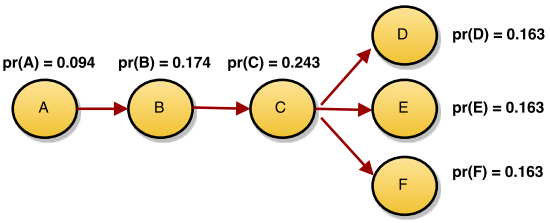
3.1 Inferring Graph Hierarchy

One way to infer graph hierarchy is through PageRank [28]. The relative importance inferred by PageRank is considered as corresponding nodes' ranking score in the graph hierarchy. However, nodes' ranking scores in graph hierarchy are not always consistent with their PageRank values even in a DAG. Take Figure 4 as an example, node C has the highest PageRank value, but it is neither the highest, nor the lowest node in the actual graph hierarchy.

3.1.1 TrueSkill. TrueSkill [12] is a Bayesian skill rating system which is designed to calculate the relative skill of players from the set of generated competitions in multi-player games. Liu et al. [22, 23] introduced a two-player and no-draw version of TrueSkill to estimate question difficulty level in community question answering

[3] A directed graph is strongly connected if there is a path between all pairs of vertices. An SCC of a directed graph is a maximal strongly connected subgraph. SCCs can be detected by the work of Nuutila et al. [25].

Figure 4: Illustrating issues of using PageRank to infer graph hierarchy (damping parameter $d = 0.85$)



services. It assumes that the practical performance of each player in one game follows a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ is the average skill of the player and σ means a system's uncertainty of the estimated skill level. Intuitively, if a system learns more about the skill of one player from more data, the standard deviation σ (uncertainty) will decrease. The skill of each player will slightly change after each game in TrueSkill. The measure $\mu - 3\sigma$ is used to rank players to ensure that top ranked players are highly skilled with high certainty [12, 22, 23].

We transform a directed graph $G = (V, E)$ into a multiplayer tournament with $|V|$ players and $|E|$ competitions. For each edge $(u, v) \in E$, we consider that u loses the competition between u and v. Based on the current estimated skill levels of two players (u and v) and the outcome of a new game between them (edge (u, v)), the TrueSkill model updates the skill level μ and σ intuitively based on whether the outcome of the new competition is expected or unexpected:

- If player v has a higher skill level than u, then the outcome of edge (u, v) is expected, and it will cause small updates in skill level μ and σ.
- If player u has a higher skill level than v, then the outcome of edge (u, v) is unexpected, and it will cause large updates in skill level μ and σ.

As far as we know, we are the first one to consider the problem of inferring graph hierarchy as a competition problem. A node v's ranking score in the graph hierarchy inferred by TrueSkill is defined as:

$$f_{ts}(v) = \mu_v - 3\sigma_v \tag{1}$$

3.1.2 Social Agony. Social agony proposed by Gupte et al. [11] assumes that the existence of a link indicates a rank recommendation. A link $u \Rightarrow v$ indicates a recommendation of v from u. If there is no reverse link from v to u, it could indicate that v is higher up in the hierarchy than u. It's assumed that in social networks such as Twitter, agony can be caused when people follow other people who are lower in the hierarchy.

Given a network $G = (V, E)$ which contains cycles, each node v has a rank $r(v)$. Higher ranking nodes are less likely to connect to lower ranking nodes. Hence, directed edges that go from higher ranking nodes are less prevalent than edges that go in reverse direction. If $r(u) > r(v)$, then edge $u \Rightarrow v$ causes agony to the node u and the amount of agony depends on the difference between their ranks.

There are many choices for defining agony. The simplest way is to define a constant value for any edge that violates the hierarchy. If it is set as a constant 1, then this problem is equivalent to a minimum feedback arc set problem as we discussed earlier. A more practical variant is to penalize the violating edges by the severity of their violation, which means that edges that respect the hierarchy receive a penalty of 0 and penalty increase linearly as the violation becomes more severe. Gupte et al. [11] defined the agony to u caused by edge (u, v) is equal to $max(r(u) - r(v) + 1, 0)$.

The agony in the network relative to the ranking r is the sum of agony on each edge: $A(G, r) = \sum_{(u,v)\in E} \max(r(u) - r(v) + 1, 0)$. Since nodes typically minimize their agony, the problem is changed to find a ranking r that minimize the total agony in the graph: $A(G) = \min_{r \in Rankings}(\sum_{(u,v)\in E} \max(r(u) - r(v) + 1, 0))$.

Gupte et al. [11] provided an $O(nm^2)$ algorithm to minimize the agony of the graph, which has n vertices and m edges. A faster discovery algorithm with the computational complexity of $O(m^2)$ was introduced by Tatti [36, 37].

A node v's ranking score in the graph hierarchy inferred by social agony is defined as:

$$f_{agony}(v) = r(v) \tag{2}$$

3.2 Strategies to select violation edges

In this section, we will introduce strategies to select edges to remove and break cycles in the graph.

Our first strategy is to select the edge which violates the graph hierarchy the most in a simple cycle (a closed path where no node appears twice, except that the first and last node are the same). Consider a simple cycle $s = (v_1, v_2, ..., v_l, v_1)$, and let $i = argmax(f(v_i) - f(v_{(i+1)\%l}))$, then edge $(v_i, v_{(i+1)\%l})$ is the one which violates the graph hierarchy the most and will be added to the set of edges to be removed. Note that removal of one edge in a simple cycle may break many other cycles at the same time, since the removed edge could be a part of multiple simple cycles. Hence it is necessary to track all simple cycles that the target edge is involved in. A major challenge in using this strategy is the time complexity to find all simple cycles in a directed graph. The time complexity for finding all simple cycles introduced by Donald [16] is $O((|V| + |E|)(|C| + 1))$ for $|V|$ nodes, $|E|$ edges and $|C|$ simple cycles. It can be computationally expensive for large graphs.

To be more efficient, we propose to simplify the input graph before using any strategy for selecting edges to break cycles. The simplification includes removing self-loops, dividing the graph into SCCs, and then dropping the trivial SCCs (a trivial SCC consists of a single node). For a non trivial SCC scc_i, several heuristic solutions to reduce scc_i to a DAG are proposed as follows:

- *Forward*: Select the node v which has the *highest* ranking score $f(v)$ in scc_i and then remove its all *out* edges in scc_i: $\{(v, u), \forall (v, u) \in scc_i\}$.
- *Backward*: Select the node v which has the *lowest* ranking score $f(v)$ in scc_i and then remove its all *in* edges in scc_i: $\{(u, v), \forall (u, v) \in scc_i\}$.
- *Greedy*: Select the edge which violates the hierarchy the most to remove. The violation of hierarchy on an edge (u, v) is defined as $max(f(u) - f(v), 0)$.

We iteratively use the above simplification processes and heuristic solutions to break cycles in the graph until the graph becomes

a *DAG*. It is easy to notice that processes for each SCC are independent from each other, and they can be parallelized to improve efficiency.

Since each node's hierarchical ranking score can be inferred by TrueSkill and Social Agony, we have 6 combinations to remove cycle edges:

- TS_G: Use TrueSkill (*TS*) to infer the graph hierarchy, and strategy *Greedy* is applied to remove cycle edges.
- TS_B: Same as TS_G, except strategy *Backward* is applied to remove cycle edges.
- TS_F: Same as TS_G, except strategy *Forward* is applied to remove cycle edges.
- SA_G: Same as TS_G, except use Social Agony (*SA*) to infer the graph hierarchy.
- SA_B: Same as TS_B, except use *SA* to infer graph hierarchy.
- SA_F: Same as TS_F, except use *SA* to infer graph hierarchy.

We use a voting schema (*H_Voting*) to ensemble the above 6 approaches for breaking cycles in a graph. For each cycle edge e, its voting score is $\sum_m(I_m(e))$, where $m \in \{TS\_G, TS\_F, TS\_B, SA\_G, SA\_F, SA\_B\}$ and $I_m(\cdot)$ is an indicator function. If edge e is removed by method m, $I_m(e) = 1$, otherwise $I_m(e) = 0$. Thus, *H_Voting* selects the edge with the highest voting score for removal.

4 EXPERIMENTS

4.1 Datasets

In this section we describe the datasets used in our experiments. We experiment not only with real-world datasets but also synthetic (random) graphs in order to demonstrate the robustness of our approach.

We use the following real-world graphs:

- *arXiv*: The Arxiv HEP-PH citation graph[4] is extracted from arXiv[5] and covers all citations from Jan. 1993 to Arpril 2003. If a paper i cites paper j, the graph contains a directed edge from i to j.
- *EU_Email*: The EU email community network graph[6] is generated using email data from a large European research institution. If a node i sent at least one message to j, then there is a directed edge from i to j.
- *Web_Google*: The Google web graph[7] is generated by representing web pages as nodes and hyperlinks between web pages as directed edges.
- *Wiki_Vote/Wiki_Talk*: Wikipedia vote/talk network graph[8] contains all the Wikipedia voting/talk data from the inception of Wikipedia till Jan. 2008. If user i voted/edited the talk page on/of user j, there is a directed edge from i to j in the graph.
- *Stackoverflow_Q2A*: The Stack Overflow network[9] contains interactions between users and questions on the stack exchange web site Stack Overflow. If user j has answered user i's question, there is a directed edge from i to j in the Stackoverflow_Q2A graph.
- *DBP_2014, DBP_2015, DBP_2016*: These three category graphs from DBpedia[10] are extracted from relationships of categories published in 2014, Oct. 2015, and April. 2016 respectively. If i is a sub-category of category j, there is a directed edge from i to j in corresponding category graph.

In addition, we also consider two graph datasets that have no cycle edges: a patent citation graph (*Cit-Patents*)[11] and the NCBI taxonomy graph (*NCBI-Taxo*) [12]. The Cit-Patent data set spans from January 1, 1963 to December 30, 1999, and includes all the utility patents granted during that period. In the Cit-Patent graph, if a patent i cites patent j, there is a directed edge from i to j. In NCBI-Taxo graph, there is a directed edge from more specific nodes to more general nodes. There is a self loop for the root node. Hence the out-degree of each node in NCBI-Taxo graph is 1.

Condensation graphs: For graphs containing cycles such as *arXiv*, *EU_Email*, etc., we use their corresponding **condensation graphs** in our experiments. The condensation graph CG of graph G is a cycle-free graph that is generated by contracting each SCC in G to a single node. Statistics of these condensation graphs are shown in Table 1.

Table 1: Statistics of Datasets

Dataset	# nodes	# edges
Cit-Patents	3,774,768	16,518,948
NCBI-Taxo	1,553,020	1,553,019
arXiv	20,085	130,469
EU-Email	230,795	223,004
Web_Google	371,603	519,304
Wiki_Vote	5,816	19,540
Gnutella	48,438	55,349
Wiki_Talk	2,394,385	5,021,410
Stackoverflow_Q2A	2,021,984	3,345,760
DBP_2014	5,502,627	20,854,028
DBP_2015	6,092,789	24,173,109
DBP_2016	6,263,925	25,211,684

Random Graphs: To provide evidence that our techniques are not merely exploiting structural properties of these particular real-world graphs, we also generated several random DAGs for our experiments. This follows the DAG(n, M) model from the random graph literature (see e.g., [1]). The following procedure is applied to generate a random DAG $RG = (N, M)$:

- Generate $|N|$ nodes with node ids in the range $[1, |N|]$.
- Randomly select $|M|$ pairs of nodes as edges in M. For each pair $\{u, v\}$, add (u, v) to M if $id(u) < id(v)$ and (v, u) otherwise.
- Randomly permute the node ids for each node in N, so that node ids do not imply anything about the order information.

[4] https://snap.stanford.edu/data/cit-HepPh.html
[5] https://arxiv.org/
[6] https://snap.stanford.edu/data/email-EuAll.html
[7] https://snap.stanford.edu/data/web-Google.html
[8] https://snap.stanford.edu/data/wiki-Vote.html and https://snap.stanford.edu/data/wiki-Talk.html
[9] https://snap.stanford.edu/data/sx-stackoverflow.html
[10] http://downloads.dbpedia.org/
[11] https://snap.stanford.edu/data/cit-Patents.html
[12] https://www.ncbi.nlm.nih.gov/taxonomy

4.2 Experimental Setup

Since there are few large real taxonomy graphs with ground truth publicly available, we consider the following set up to evaluate the different approaches: We consider a large real or synthetic DAG and introduce cycles in it by inserting edges that violate the partial ordering induced by the DAG. The goal for the various approaches is to identify the set of edges that were inserted. In particular, we evaluate the performance of different approaches by considering the set of newly inserted cycle-introducing edges as the ground truth and computing *precision*, *recall*, and *F1-score* with respect to this ground truth.

To introduce cycles, we repeatedly perform the following step: randomly select a node pair (u, v) and if u can reach v in the input DAG, then we insert the edge (v, u) into the cycle-introducing edge set T. In some experiments, we also constrain the shortest path length of $u \rightarrow v$ to be no larger than a threshold d. Once we have the required number of edges in T, we insert these edges into the input DAG. Hence, edges in T are labeled as *noisy edges* which can be used for evaluation.

Baselines to Remove Cycle Edges:

Three baseline approaches are used in our experiments:

- *DFS*: use DFS to detect and remove back edges
- *PR*: use PageRank to infer graph hierarchy, and strategy *Greedy* is applied to break cycles.
- *MFAS*: a local greedy implementation of minimum feedback arc set problem, as described in section 2.

In addition to the precision, recall and F1-score obtained using the above setup, we also consider auxiliary measures such as the number of edges removed to make the graph acyclic.

4.3 Experimental Results

First, we consider the general case where there are no constraints on the length (d) of the cycle. Figure 5 presents our results on random DAGs (with $RG(n, m)$ representing a random DAG with n nodes and m edges, where 1K is used as shorthand for 1000). We inserted 1500[13] random edges into these graphs to create cycles[14], and the goal is to identify and remove these edges in order to break the cycles. We observe that the different approaches achieve very different precision, recall and overall F1 score. Our proposed voting approach, *H_Voting*, achieves the best F1 score across the entire range of different random DAGs. Note that our approach is much more accurate for these graphs than the traditional heuristics based on *DFS* and *MFAS* on random DAGs.

Next, we consider the real-world graphs, and, as before, insert $1,500$ random edges to create cycles. As shown in Figure 6, there is no consistent winning strategy. While most approaches based on TrueSkill and Social agony achieve fairly similar performance on *Cit-Patents*, TS-B achieves the best F1 score on the tree-like NCBI taxonomy graph[15]. We note that *H_Voting* achieves the best F1 score (around 0.9) on the *arXiv* condensation graph and is among the top-3 over all settings, both in terms of precision and F1. In contrast, the F1 score of *DFS* based heuristic is 0.12 for *Cit-Patents*, 0.17 for *NCBI-Taxo* and 0.02 for *arXiv*.

[13] We have tested the performance on a varying number of edges as shown in Section 4.5. In the interest of space, we report only the performance for 1500 extra edges. Other settings yield similar results.

[14] The maximum number of big SCCs generated is 1500

[15] After removal of the only self-loop edge.

4.4 Number of Edges to be Removed

In addition to precision, recall and F1 score, another important performance measure is the number of edges removed. Although there is no empirical evidence that removing fewer edges causes less damage to the logical hierarchy of the ontological relation, we still want as few edges to be removed as possible. In particular, this is the measure that is directly optimized by the minimum feedback arc set (*MFAS*) problem.

For brevity, we only report the number of edges that are removed from graphs $RG(3K, 15K)$, $RG(30K, 150K)$, and $RG(10K, 150K)$ in Table 2. In these cases, 1500 random edges were inserted to introduce cycles. We note that TS_G, SA_G, and *H_Voting* remove fewer edges compared to other approaches. Furthermore, the number of removed edges is close to 1500, which is the number of edges in the ground truth. Interestingly, the number of edges removed by TS_G, SA_G, and *H_Voting* is considerably smaller than by the greedy heuristic for *MFAS*, which is directly minimizing the number of removed edges. The results in other settings are similar. The corresponding precision, recall and F1 scores are shown in Figure 5, and indicate that strategies that remove fewer edges also have higher F1 scores.

Table 2: # edges to be removed

# edges to be removed	RG(3K,15K)	RG(30K,150K)	RG(10K,150K)
DFS	6,057	17,587	54,774
PR	3,337	5,155	8,478
MFAS	2,423	3,378	4,584
TS_B	2,431	3,622	3,865
TS_F	2,307	3,232	3,736
SA_F	2,175	2,077	2,707
SA_B	2,193	1,881	2,566
TS_G	1,860	1,544	1,597
SA-G	1,691	1,506	1,531
H_Voting	1,649	1,502	1,513

4.5 Sensitivity to Number of Noisy Edges

Next, we test the sensitivity of H_Voting to the number of noisy edges in the hierarchical relationship. For this, we consider small ($RG(3K, 15K)$), medium ($RG(3K, 30K)$) and a large ($RG(3K, 45K)$) random DAGs. Figure 7 shows how the precision ("-p" in legend), recall ("-r") and F1 ("-f1") scores vary on these graphs as the fraction of noisy edges is increased (e.g., a fraction of 0.1 in the x-axis corresponds to adding $1.5K$, $3K$, and $4.5K$ noisy edges in $RG(3K, 15K)$,$RG(3K, 30K)$, and $RG(3K, 45K)$, respectively, for introducing cycles). As expected, the accuracy scores decrease as the fraction of incorrect edges is increased. But more importantly, we find that the accuracy of our approaches becomes more robust to noise as the graph size increases. For instance, when 30% ($13.5K$) extra noisy edges are added to $RG(3K, 45K)$, *H_Voting* is still able to achieve a F1 score of around 0.8 and recall of around 0.9. This gives us confidence that in large real-world graphs, *H_Voting* can accurately identify the edges to remove, even in scenarios with large amounts of noise.

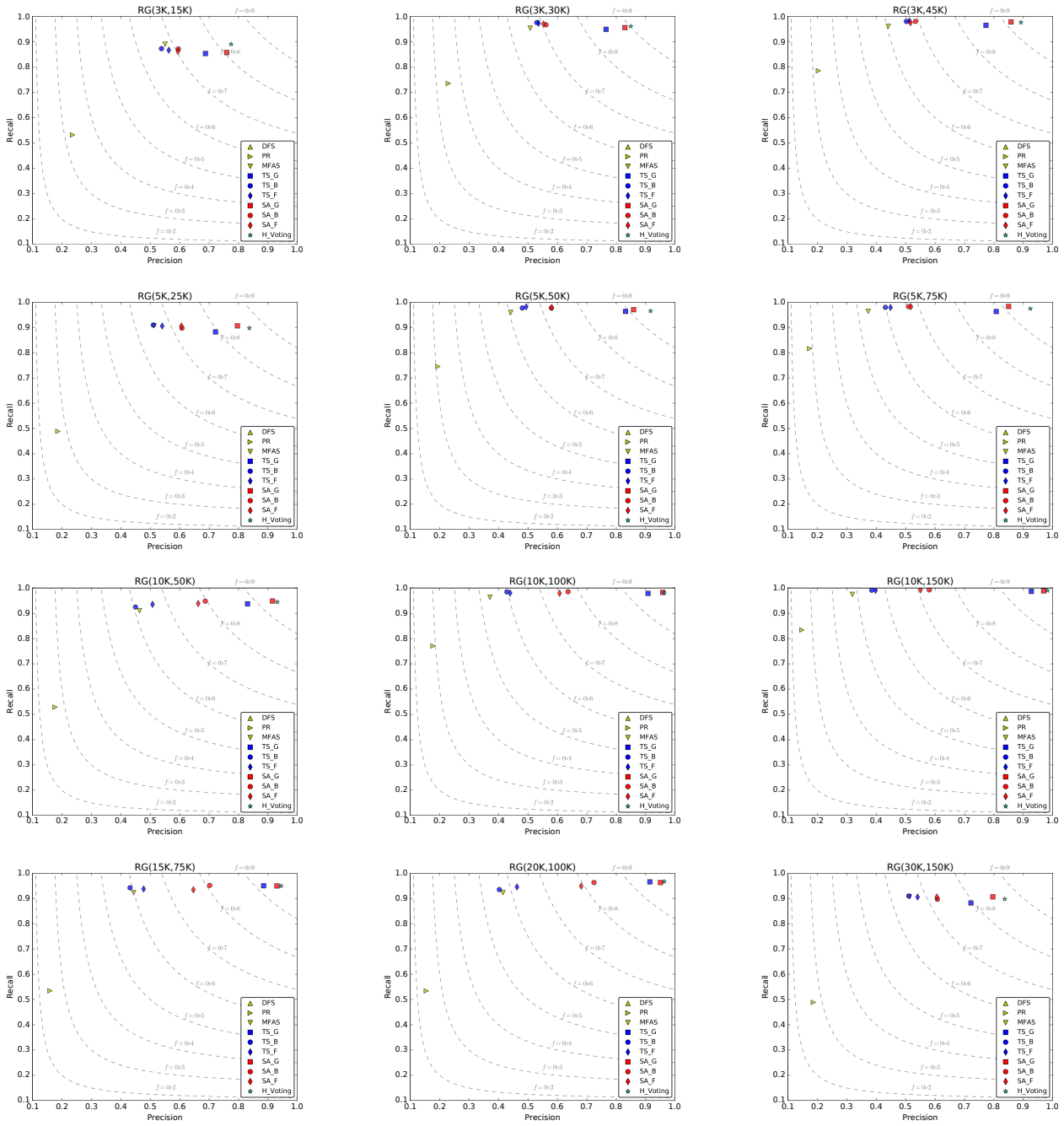


Figure 5: Performance (precision,recall,f-1 score) on different sizes of random generated graphs (path length control parameter d is unlimited). We can observe that our proposed voting approach, *H_Voting*, achieves the best F1 score across the entire range of different random DAGs.

4.6 Special case of constrained cycle length

To further understand why the TrueSkill and Social agony based approaches are more accurate than the traditional heuristics based on *MFAS*, *DFS* or *Pagerank*, we consider the special case in which cycles are constrained to be of length at most two. We consider a range of real-world graphs and insert 1,500 random edges to introduce cycles. Note that it is easy to obtain high accuracy in this setting, as even randomly selecting one of the two edges from every simple cycle achieves an expected 0.5 precision and recall. As expected, we observe from Figure 8 that generally, most approaches achieve

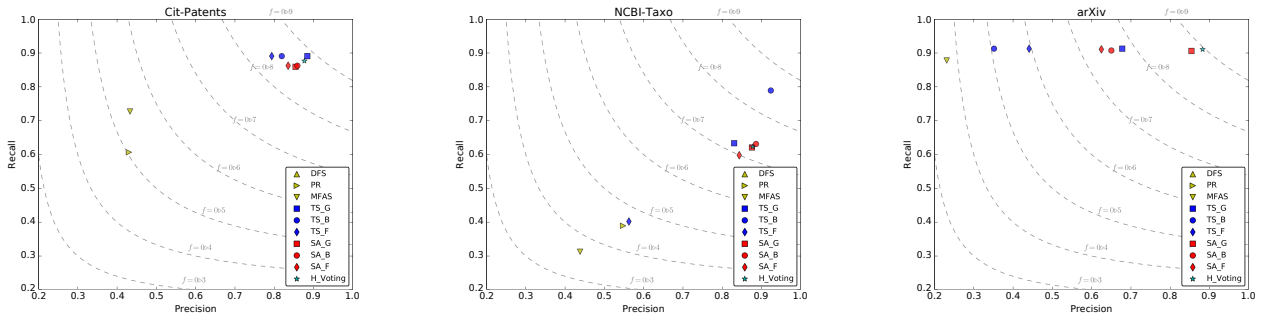
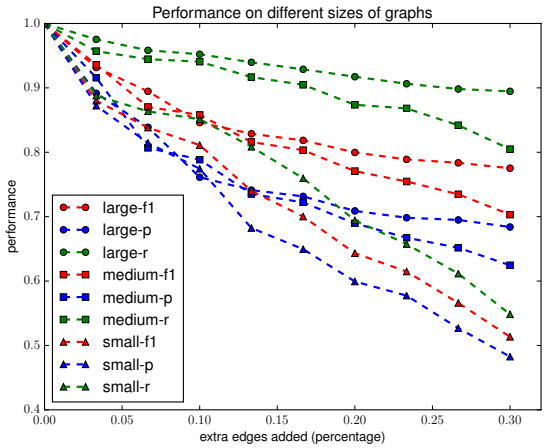


Figure 6: Performance (precision,recall,f-1 score) on different datasets' corresponding graphs (condensation graph for *arXiv*); path length control parameter d is unlimited

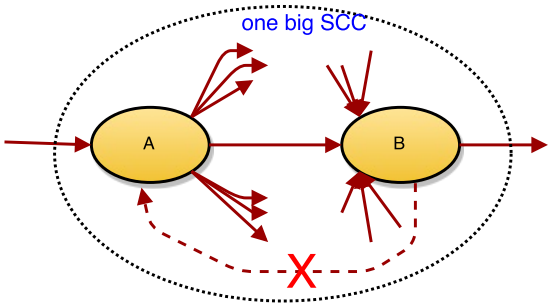
Figure 7: Sensitivity to Extra Edges Added



high precision and recall values. However, even here we find that sometimes the accuracy of *DFS*, *Pagerank* and *MFAS* approaches can be quite poor (e.g., consider *DFS* and *Pagerank* on *Web_Google* graph, *DFS* on *Wiki_Talk* and *PR* and *MFAS* on *Stackoverflow_Q2A* graphs). On the other hand, TrueSkill, Social Agony and *H_Voting* approaches consistently achieve promising F1 scores.

Interestingly, there are a few graphs in this special setting, where *MFAS* based greedy heuristic does outperform the other approaches slightly (e.g., *Web_Google*) or performs as well as the other approaches. To understand this, consider Figure 9, where edge (B, A) is the randomly added edge that generates a SCC in the corresponding graph. A node i's ratio value is defined as $r(i) = \frac{d_i^{in}}{d_i^{out}}$. In this example $r(B) \gg 1$ and $r(A) \ll 1$. Then edge (B, A) will be selected as the edge to be removed based on the local greedy implementation of *MFAS*. The average ratio value of source nodes such as B in Figure 9 is 14, 13.23, and 13.97 in *EU-Email*, *Web_Google* and *Wiki_Vote* graph respectively. And the average ratio value of target nodes such as A in Figure 9 is 0.076, 0.85, and 0.0664 in *EU-Email*, *Web_Google* and *Wiki_Vote* graph respectively. As a result, the greedy *MFAS* heuristic correctly picks up edge (B, A) for removal in many of these graphs. However, in graphs with larger simple cycles, these heuristics suffer from very poor precision (as already noted in Section 4.3).

Figure 9: A toy scenario in which MFAS can outperform other strategies.



4.7 Performance on Wikipedia Category Graph

We apply our approach to remove cycle edges in the Wikipedia category graphs. Note that this is a graph extracted from a 2014 snapshot of the popular skos:broader relationships in DBpedia. First, we note that even though skos:broader captures the hierarchical specific/general relationship, it has a large number of cycles forming many SCCs (c.f. Table 3). This is because it is created in a crowd-sourced way and its creators often misinterpret the granularity of concepts. For instance, the concept United Nations can refer to its headquarter in New York or it can refer to the intergovernmental organization, creating a cycle United Nations → New York → United States of America → United Nations. The results are similar for later snapshots of Wikipedia categories.

Table 3: Statistics of big SCCs in DBpedia

DBpedia	2014
# SCCs	534
# nodes in SCCs	8,939
# edges in SCCs	24,178
# nodes in the biggest SCC	6,741
# edges in the biggest SCC	20,979

Since we don't have a good ground truth for edges to remove in this dataset (there are a large number of cycles destroyed, new

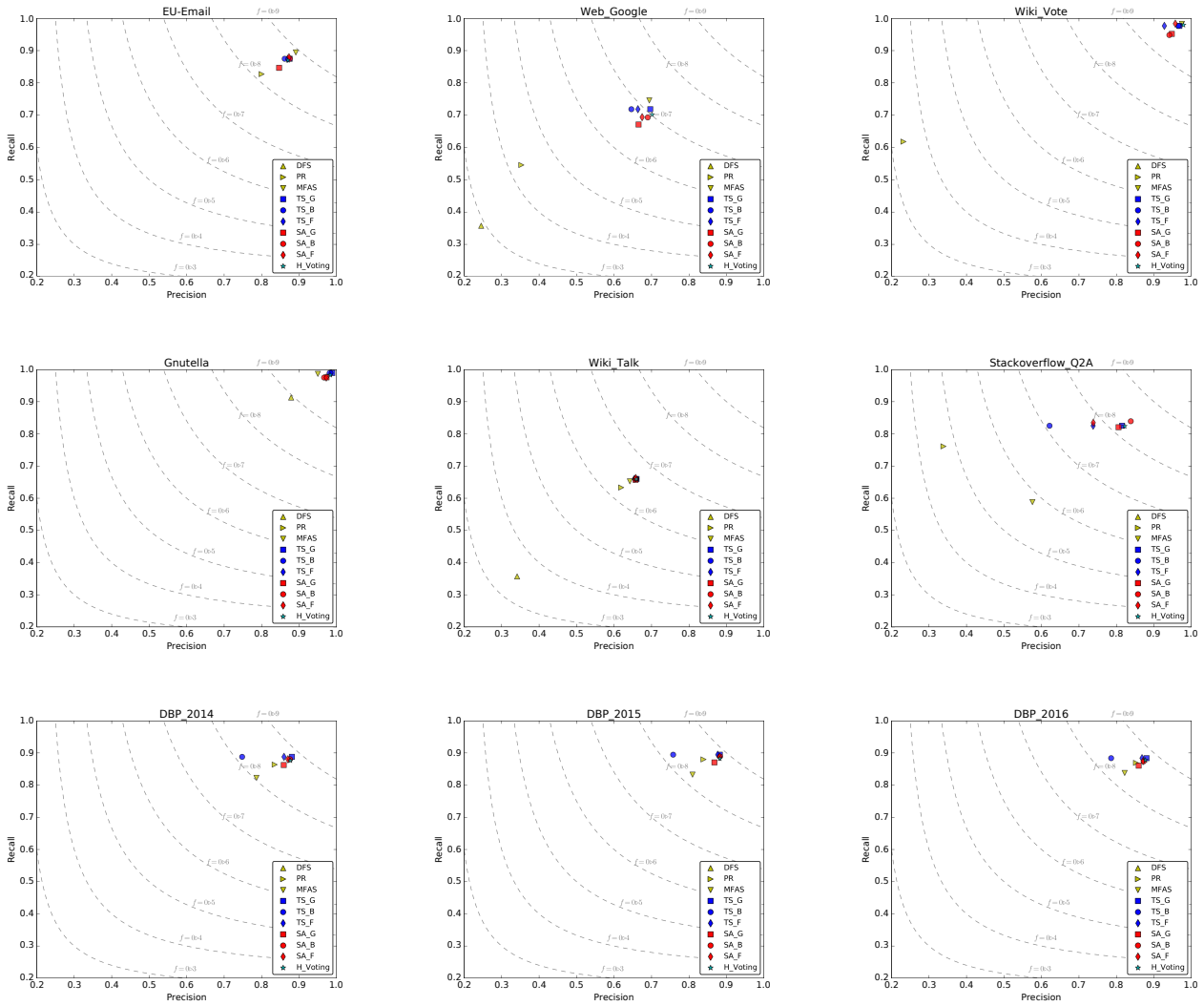


Figure 8: Performance (precision,recall,f-1 score) on different datasets' corresponding condensation graphs (path length control parameter $d = 2$)

cycles created and many new nodes and edges created from one snapshot to another), we focus on the number of edges that are removed by different approaches. We observe from Table 4 that, similar to the results on random DAGs in Section 4.4, *H_Voting* and SA_G break cycles in *DBP_2014* graph by removing considerably fewer edges compared to other approaches.

Table 4: # edges to be removed

# edges to be removed	DBP_2014 Category Graph
MFAS	4,075
PR	3,920
DFS	3,602
TS_F	3,030
TS_B	2,501
TS_G	2,479
SA_F	1,737
SA_B	1,730
H_Voting	1,713
SA_G	1,672

5 CONCLUSION

In this paper, we address the problem of breaking cycles while preserving the logical structure (hierarchy) of the directed graph as much as possible. We propose approaches that explicitly focus on inferring the graph hierarchy using TrueSkill and Social Agony. We leverage this inferred hierarchy using an ensemble approach to identify the edges to be removed. We show that our approaches

achieve significantly better accuracy compared to the traditional heuristics based on DFS and MFAS and at the same time, they are fast, scalable and fully automated. Thus, they can support a large and growing number of applications that rely on clean ontological knowledge bases representing hierarchical relations.

Future work. In this study, the issue of breaking cycles from directed graphs is addressed from a heuristic perspective. An alternative is to consider model based approaches to predict the edge in a SCC that has the highest probability of being removed. The required features for the prediction model can be extracted from graph embedding methods, such as node2vec[10], or low rank representations of adjacency matrices, decomposed by matrix factorization (a widely used technique in recommender systems [4, 19, 32, 40, 41]). Furthermore, other measures, such as deviation to dominant structural role, and deviation to transitive closure, may also prove to be very useful both for the heuristic and model-based approaches. However, significant effort is required to compute these measures for large graphs in a fast and scalable way.

Acknowledgments This work is supported by the National Science Foundation of United States under grant CCF-1645599 and IIS-1550302. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Deepak Ajwani and Tobias Friedrich. 2010. Average-case analysis of incremental topological ordering. *Discrete Applied Mathematics* 158, 4 (2010), 240–250.

[2] Ali Baharev, Hermann Schichl, and Arnold Neumaier. 2015. An exact method for the minimum feedback arc set problem. (2015).

[3] Shuvra S Bhattacharyya, Praveen K Murthy, and Edward A Lee. 1996. *Software Synthesis from Dataflow Graphs*. Vol. 360. Springer Science & Business Media.

[4] Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. 2017. Learning to Recommend Accurate and Diverse Items. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 183–192.

[5] James J Cimino. 1998. Auditing the unified medical language system with semantic methods. *Journal of the American Medical Informatics Association* 5, 1 (1998), 41–51.

[6] Peter Eades and Xuemin Lin. 1995. A new heuristic for the feedback arc set problem. *Australasian Journal of Combinatorics* (1995), 15–25.

[7] Peter Eades, Xuemin Lin, and William F Smyth. 1993. A fast and effective heuristic for the feedback arc set problem. *Inform. Process. Lett.* 47, 6 (1993), 319–323.

[8] G. Even, J. (Seffi) Naor, B. Schieber, and M. Sudan. 1998. Approximating Minimum Feedback Sets and Multicuts in Directed Graphs. *Algorithmica* 20, 2 (1998), 151–174.

[9] Marco Fossati, Dimitris Kontokostas, and Jens Lehmann. 2015. Unsupervised Learning of an Extensive and Usable Taxonomy for DBpedia. In *Proceedings of the 11th International Conference on Semantic Systems (SEMANTICS '15)*. 177–184.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.

[11] Mangesh Gupte, Pravin Shankar, Jing Li, S. Muthukrishnan, and Liviu Iftode. 2011. Finding Hierarchy in Directed Online Social Networks. In *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*. New York, NY, USA, 557–566.

[12] Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. TrueSkill™: A Bayesian Skill Rating System. In *Advances in Neural Information Processing Systems (NIPS)*, P. B. Schölkopf, J. C. Platt, and T. Hoffman (Eds.). 569–576.

[13] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence* 194 (2013), 28–61.

[14] Chia-Jui Hsu and Shuvra S Bhattacharyya. 2007. *Cycle-breaking techniques for scheduling synchronous dataflow graphs*. Technical Report.

[15] Chia-Jui Hsu, Ming-Yung Ko, Shuvra S Bhattacharyya, Suren Ramasubbu, and José Luis Pino. 2007. Efficient simulation of critical synchronous dataflow graphs. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 12, 3 (2007), 21.

[16] Donald B Johnson. 1975. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.* 4, 1 (1975), 77–84.

[17] Viggo Kann. 1992. *On the approximability of NP-complete optimization problems*. Ph.D. Dissertation.

[18] Richard M. Karp. 1972. *Reducibility among Combinatorial Problems*. Boston, MA, 85–103.

[19] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug 2009), 30–37.

[20] Javier Lacasta, Javier Nogueras-Iso, and Francisco Javier Zarazaga-Soria. 2010. *Terminological Ontologies - Design, Management and Practical Applications*. Semantic Web and Beyond: Computing for Human Experience, Vol. 9. Springer.

[21] Wooyoung Lee and Dale F Rudd. 1966. On the ordering of recycle calculations. *AIChE Journal* 12, 6 (1966), 1184–1190.

[22] Jing Liu, Young-In Song, and Chin-Yew Lin. 2011. Competition-based User Expertise Score Estimation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. New York, NY, USA, 425–434.

[23] Jing Liu, Quan Wang, Chin-Yew Lin, and Hsiao-Wuen Hon. 2013. Question Difficulty Estimation in Community Question Answering Services. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 85–90.

[24] Bodenreider Olivier Mougin Fleur. 2005. Approaches to Eliminating Cycles in the UMLS Metathesaurus: Nave vs. Formal. *American Medical Informatics Association Annual Symposium Proceedings* (2005), 550–554.

[25] Esko Nuutila and Eljas Soisalon-Soininen. 1994. On Finding the Strongly Connected Components in a Directed Graph. *Inf. Process. Lett.* 49, 1 (Jan. 1994), 9–14.

[26] Bodenreider Olivier. 2001. Circular Hierarchical Relationships in the UMLS: Etiology, Diagnosis, Treatment, Complications and Prevention. *Proceedings of the American Medical Informatics Association Symposium* (2001), 57–61.

[27] Tatiana Orenstein, Zvi Kohavi, and Irith Pomeranz. 1995. An optimal algorithm for cycle breaking in directed graphs. *Journal of Electronic Testing* 7, 1 (1995), 71–81.

[28] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.

[29] Domenico M Pisanelli, Aldo Gangemi, and Geri Steve. 1998. An ontological analysis of the UMLS Metathesaurus.. In *Proceedings of the AMIA symposium*. American Medical Informatics Association, 810.

[30] Youssef Saab. 2001. A Fast and Effective Algorithm for the Feedback Arc Set Problem. *Journal of Heuristics* 7, 3 (May 2001), 235–250.

[31] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 697–706.

[32] Jiankai Sun, Shuaiqiang Wang, Byron J. Gao, and Jun Ma. 2012. Learning to Rank for Hybrid Recommendation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. 2239–2242.

[33] Osma Suominen and Eero Hyvönen. 2012. Improving the Quality of SKOS Vocabularies with Skosify. In *Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12)*. 383–397.

[34] Osma Suominen and Christian Mader. 2014. Assessing and Improving the Quality of SKOS Vocabularies. *Journal on Data Semantics* 3, 1 (2014), 47–73. DOI: http://dx.doi.org/10.1007/s13740-013-0026-0

[35] Roberto Tamassia. 2007. *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC.

[36] Nikolaj Tatti. 2014. *Faster Way to Agony Discovering hierarchies in directed graphs*. Berlin, Heidelberg, 163–178.

[37] Nikolaj Tatti. 2015. Hierarchies in Directed Networks. In *2015 IEEE International Conference on Data Mining*. 991–996.

[38] Hahn Udo and Stefan Schulz. 2004. Boosting the Medical Knowledge Infrastructurefi!?A Feasibility Study on Very Large Terminological Knowledge Bases. *Proc Symp on Engineering of Intelligent Systems* (2004).

[39] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[40] Shuaiqiang Wang, Jiankai Sun, Byron J. Gao, and Jun Ma. 2012. Adapting Vector Space Model to Ranking-based Collaborative Filtering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. 1487–1491.

[41] Shuaiqiang Wang, Jiankai Sun, Byron J. Gao, and Jun Ma. 2014. VSRank: A Novel Framework for Ranking-Based Collaborative Filtering. *ACM Trans. Intell. Syst. Technol.* 5, 3, Article 51 (July 2014), 24 pages.

[42] Torsten Zesch and Iryna Gurevych. 2007. Analysis of the Wikipedia Category Graph for NLP Applications. In *Proceedings of the TextGraphs-2 Workshop (NAACL-HLT)*. Association for Computational Linguistics, Rochester, 1–8.