iHypR: Prominence Ranking in Networks of Collaborations with Hyperedges [1]

Sibel Adalı, Department of Computer Science, Rensselaer Polytechnic Institute
Malik Magdon-Ismail, Department of Computer Science, Rensselaer Polytechnic Institute
Xiaohui Lu, Department of Computer Science, Rensselaer Polytechnic Institute

We present a new algorithm called iHypR for computing prominence of actors in social networks of collaborations. Our algorithm builds on the assumption that prominent actors collaborate on prominent objects, and prominent objects are naturally grouped into prominent clusters or groups (hyperedges in a graph). iHypR makes use of the relationships between actors, objects and hyperedges to compute a global prominence score for the actors in the network. We do not assume the hyperedges are given in advance. Hyperedges computed by our method can perform as well or even better than 'true' hyperedges. Our algorithm is customized for networks of collaborations, but it is generally applicable without further tuning. We show, through extensive experimentation with three real life data sets and multiple external measures of prominence, that our algorithm outperforms existing well known algorithms. Our work is the first to offer such an extensive evaluation. We show that unlike most existing algorithms, the performance is robust across multiple measures of performance. Further, we give a detailed study of the sensitivity of our algorithm to different data sets and the design choices within the algorithm that a user may wish to change. Our paper illustrates the various trade-offs that must be considered in computing prominence in collaborative social networks.

Categories and Subject Descriptors: H.3.1 [**Content Analysis and Indexing**]; H.3.3 [**Information Search and Retrieval, Relevance Feedback**]; J.4 [**Social and Behavioral Sciences**]

General Terms: Algorithms, Measurement, Performance

Additional Key Words and Phrases: Prominence, Collaboration, Social networks

ACM Reference Format:
Adalı, S., Magdon-Ismail, M., and Lu, X. 2012. iHypR: Prominence Ranking in Networks of Collaborations with Hyperedges. ACM Trans. on Knowl. Disc. from Data 1, 1, Article 1 (March 2012), 32 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

A social collaborative network contains actors (for example authors) who collaborate on creating objects (for example papers). In today's online world, social collaborative

[1]Note: A preliminary version of this paper appears in ICWSM 2011 under the title "Prominence Ranking in Graphs with Community Structure".

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.
Author's addresses: Sibel Adalı, Department of Computer Science, Rensselaer Polytechnic Institute; Malik Magdon-Ismail, Department of Computer Science, Rensselaer Polytechnic Institute; Xiaohui Lu, Department of Computer Science, Rensselaer Polytechnic Institute.

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

networks surround us, perhaps the most famous being Wikipedia where the authors are editors and the objects they collaborate on are Wikipedia articles. Similarly, authors in academic social networks collaborate to create papers; bloggers collaborate to create blogs; actors collaborate to create movies; employees in a firm ‘collaborate’ to create email threads. In general, such social collaborative networks can be represented by a bipartite graph as illustrated in Figure 1. The actors and objects are the nodes,

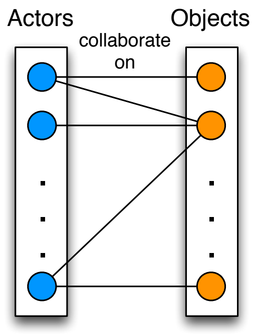

Fig. 1. Illustration of a social collaborative network.

forming the two partitions in the graph and an edge from an actor to an object means that the actor collaborated on that object. An actor can collaborate on multiple objects and multiple actors can collaborate on the same object.

We address the problem of ranking the actors according to prominence; the input is the social collaborative network. The rationale for why one should be able to infer prominence from a social collaborative network is that the objects have some value. Hence, even a crude estimate such as just the number of objects an actor has collaborated on (the actor node’s degree) can be a useful measure of prominence. We can do better than such crude estimates because prominent actors would rather collaborate on more valuable objects. Even more is true: objects are generally viewed by an external audience (for example readers of papers; viewers of movies); hence, a prominent actor gains social status from being associated to high value objects and tries to be disassociated from low value objects. As a result, there is a tendency for those actors who collaborate on a particular object to be of comparable prominence. This basic notion is the general idea behind any algorithm which attempts to determine promince from an observed social collaborative network. In general, one starts with a crude estimate of prominence, such as node degree and then refines it.

We introduce a new algorithm called iHypR (iterative Hyperedge[2] Ranking) that ranks actors in the order of prominence in such social collaborative networks. Our algorithm is specifically designed for a network of collaborations which satisfies the following conditions:

— The collaborations of people (actors) produce objects with external value.
— The prominence of the individuals are linked to the prominence of the objects they have collaborated on.

Many of the algorithms introduced for prominence computation either consider graphs induced on only the actors or only the objects. For example actor to actor graphs [Faust

[2]In graph theory [West 2000], a hyperedge is a set of (possibly) more than two vertices as opposed to an edge which is a set of exactly two vertices

and Wasserman 1994], object to object graphs [Brin and Page 1998; Kleinberg 1999]. Some algorithms consider the actor to object graphs [Hotho et al. 2006]. Our algorithm will use group structure among the objects to compute prominence scores for actors. There are a number of algorithms that use group structure in different ways: some require setting values of several tunable parameters specific to the data [Balmin et al. 2004; Sun et al. 2009a]; others operate on data sets that do not correspond to collaborations [Hotho et al. 2006; Bao et al. 2007] and employ different hypotheses [Sun et al. 2009b]. For example, rankclus and netclus [Sun et al. 2009b] compute a prominence ranking in different *actor* communities separately (the rationale is that actors in different communities may be incomparable); these communities must be specified ahead of time. The group structure among objects which are the basis for our algorithm may be specified by hyperedges, which we now describe.

1.1. Hyperedges

It is often the case that the objects in the collaborative network fall into natural groups. For example, papers belong to conferences and fields; movies into genres and sequels or trilogies, etc. A group of objects is a hyperedge. Prominent, high value objects tend to belong to prominent hyperedges. If we augment the social collaborative network with hyperedges, the resulting picture of the network is illustrated in Figure 2. Actors are linked to objects that they create and objects are linked together in hyperedges. As an example, consider a network of academic collaborations. The actors, i.e. researchers, collaborate on papers (objects). Objects are naturally placed in hyperedges like conferences or journals. High quality conferences and prominent individuals tend to have high value publications.

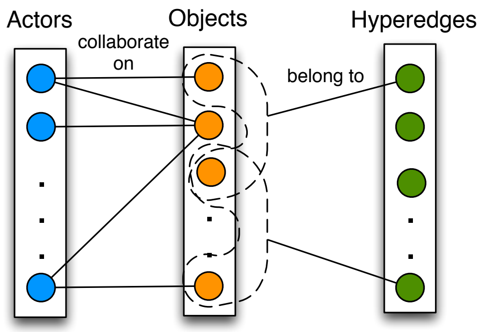

Fig. 2. A collaborative social network with hyper-edges. For example, the actors can be authors; the objects are papers; and, the hyperedges are conferences. A hyperedge is a collection of objects.

Hyperedges can help with computing prominence for two reasons. First, is the *informative* aspect; an object in a hyperedge (group) with other high value objects will likely have high value. Thus, the hyperedges allow one to impute the value of an object from the values of other objects in the group. Second, is the *noise dampening* effect. An object's value is hard to evaluate and may not be a direct consequence of the actors who collaborate on it: for academic collaborations on papers one might argue that it is the authors who are solely responsible for the quality of a paper; on the other hand, the actors alone are not responsible for the quality of a movie (many other ingredients such as special effects etc. have an impact). Thus, a single object's value is a noisy indicator of the prominence of an actor, and a group of objects (a hyperedge) can give a more stable estimate.

We emphasize that the hyperedges are not specified in the data. In some networks, natural hyperedges exist; for example, academic papers are naturally grouped into conferences. In general, such natural hyperedges are not available. Our algorithm iHypR *does not need the hyperedges to be prespecified*. Algorithms which use some form of hyperedge structure typically assume that these hyperedges are prespecified. For example [Sun et al. 2009a; Sun et al. 2009b] assumes actor communities are given. Our approach is based on inferring the hyperdeges.

1.2. Our Contributions

We give a method for ranking actors in an actor-object collaborative network. One important aspect of our algorithm is the use of hyperedge structure among the objects. This hyperedge structure is computed within the algorithm using an external clustering algorithm, and we show results with two algorithms that work well for this purpose. Furthermore, we also show that hyperedges found by clustering algorithms actually perform better than 'true' hyperedges (for example conferences in academic networks) since the inferred hyperedges reduce the noise present in the true hyperedges.

Most algorithms for prominence computation are carefully tuned to a particular network, for example [Balmin et al. 2004; Nie et al. 2005; Hotho et al. 2006; Bao et al. 2007]. Such algorithms assume a great deal of information about the underlying data which limits their applicability. There is often no discussion of how sensitive the algorithm is to this tuning. These methods are not competitive for the datasets we study here because they either suppress too much relevant data or introduce too much noise. In our algorithm there are a few design choices to be fixed. We present a default version of our algorithm which can be applied to any network without any further tuning. We use this single algorithm in all our experiments. We also study how the different design choices affect the performance.

Evaluation. Most of the work that ranks objects in multi-genre networks rely on anecdotal evidence or limited evaluation of the performance of the algorithms. Here, we use three real life datasets containining tens of thousands to millions of nodes and many different outside measures of prominence to validate our algorithm. We compare against many known algorithms and show that our algorithm is robust across many different application domains. In particular, we study a dataset of academic collaborations based on peer review (Digital Bibliography & Library Project, DBLP), a dataset of e-mail communications in a hierarchical organization (Enron dataset) and a dataset of movies from Hollywood as well as other movie industries (Internet Movie Database, IMDB). These datasets of collaborations offer us interesting case studies and we examine them in depth in our tests. The following are the unique contributions of our paper:

1. iHypR*: iterative Hyperedge Ranking*. A novel algorithm for ranking in an (actor)-(object) bi-partite collaborative social network that is based on constructing an (actor)-(object)-(hyperedge) tri-partite social network (Figure 2). iHypR is iterative and is based on the assumption that actors derive prominence from their objects and objects derive prominence from the hyperedges they belong to.
2. We demonstrate improved ranking performance when using hyperedges as compared with ignoring them. We demonstrate this by using conference venues as 'true' hyper edges in DBLP to significantly improve the ranking of DBLP authors. We measure the quality of results using a ground truth defined from citation counts. This conclusion holds not only for our algorithm but also the natural extensions of existing algorithms to our tri-partite setting. Thus, *hyperedges are important for ranking*.

3. We show that it is possible to infer the hyperedges as *overlapping clusters* of the objects. We show improved performance when using the inferred hyperedges as compared to the 'true' hyperedges (which we take to be publication venue in DBLP). *Inferred hyperedges can be more robust than proclaimed hyperedges.*
4. We study the impact of various network environments on ranking performance. Specifically, we use Enron data to explore the affect of an hierarchical or organizational structure amongst actors; and, we use IMDB data to explore the effect of intrinsic object value and capital gain for collaborating actors. We show that our algorithm significantly outperforms all other well-known algorithms for all these data sets.
5. We study how ranking performance using hypergraphs changes when we change how social ties and objects are valued. In particular we observe that the Jaccard measure, which favors a pairwise notion of social distance, performs better for clustering algorithms that have a more localized view of distance, and the Adamic/Adar measure, which favors a more global value for social distance, performs better for clustering algorithms that incorporate more global information.
6. We give extensive experimentation with real datasets using *external* measures of prominence to show that iHypR provides superior performance and is robust across a number of different criteria. This is the first large scale study of such ranking algorithms with multiple data sets and external performance criteria.

The basis for our computation of actor prominence is that objects have value and the actors are directly responsible for the value of these objects; higher prominence actors create higher value objects. This is explicitly the case in academic networks and only loosely the case in an email network (emails generally have little value and not all correspondents in an email thread are responsible for that albeit little noisy value). We present two versions of our algorithm that are applicable to each of these types of cases. Loosely speaking, represent the actor-object-hyperedge graph as $A - O - H$. The basic algorithm applicable to the case where objects have value that is the direct result of the prominence of the actors (for example academic networks) is based on this $A - O - H$ graph. When object values are more noisy and not necessarily directly reflecting the prominence of the actors (for example movie or email networks) the noise in the object values can be misleading and it is the noise dampening effect of the hyperedges that is responsible for most of the improvement. In such types of networks it is advisable to bypass the object network and only consider the 'smoothed' actor-hyperedge network $A - H$. We show results for our algorithm based on both the $A - O - H$ and $A - H$ networks. As expected the $A-O-H$ algorithm dominates in the academic network and the $A - H$ algorithm dominates in the other networks. In all networks, our algorithms dominate algorithms based on prior literature.

2. THE IHYPR ALGORITHM

In this section, we introduce an algorithm for ranking objects based on hyperedge structure. We consider a bi-partite graph $G = (V, E)$ where each node $v \in V$ has a type: either an actor or an object. We use A to denote all nodes in V of type actor, and B to denote all nodes in V of type object, where $V = A \cup B$ and $A \cap B = \emptyset$. Actors collaborate on objects and this collaboration constitutes a tie between the actor and the object. Thus, each undirected edge in E is of the form (a, b), between an actor $a \in A$ and an object $b \in A$. For example, in an academic setting actors are researchers and objects are papers and edges are from authors to papers. In the movie industry, the actors collaborate on movies as objects. In a communication data set, emails could be the objects and the sender and plus recipients are the actors 'collaborating' on the object.

A hyperedge [West 2000] is an edge that connects multiple vertices, instead of just two. Hence, a hyperpedge c connects a set of objects $c \subseteq B$ that are related to each other somehow. We would like to find a set C of hyperedges, each connecting a set of objects from B. Note that in some cases, the hyperedges may be disjoint: in other words no two hyperedges may connect to the same vertex. This is true if hyperedges represent conferences containing sets of papers (the objects). However, if hyperedges represent research areas, then they can also be overlapping. A certain paper may belong to multiple research areas. Even for conferences, there are multiple ways to define hyperedges, using different conference series (like WWW) or each specific year of a conference (WWW2012). Regardless of the underlying definition, we would like to find hyperedges that represent objects of similar prominence such as those in the same conference.

One conclusion we can draw from this discussion is that the appropriate hyperedge for a dataset may not be obvious. Furthermore, some datasets may have natural hyperedges, and some may not. In our algorithm, we do not make the assumption that hyperedges are known. We first describe how to find hyperedges. We will later examine the impact of using our algorithm as opposed to the natural hyperedges (eg. venues in DBLP such as conferences, journals and books).

2.1. Computing Hyperedges

To compute hyperedges, given $G = (V, E)$, we first construct a new graph $G_O = (V_O, E_O)$ on the objects with undirected weighted edges as follows.

$$\begin{aligned} V_O &= B \\ E_O &= \{(b_1, b_2) \mid b_1, b_2 \in V_O \text{ and } \{(a, b_1), (a, b_2)\} \subseteq E \text{ for some } a \in A\} \end{aligned}$$

That is, an edge in the object graph exists between two objects if the same author collaborated on both objects. We then define weights as the distance between two objects, denoted by $w_{aa}(b_1, b_2)$ for each edge as follows: the weight measure we choose is inspired by the Adamic-Adar measure [Adamic and Adar 2003] given for actor to actor graphs. Here, we apply it to actor-object graphs to determine an object-object distance. The distance between the objects depends on the common actors. For each common actor, the distance is larger if that actor is linked to many other objects and therefore spent ‘less energy’ on these objects (as each of the object gets less attention from the actor); if the actor links to many other objects, it means the actors link between any two is weaker. We sum this distance over all the common authors to the two objects to find the weight.

$$w_{aa}(b_1, b_2) = \frac{1}{\sum\limits_{b_1, b_2 \in \Gamma(a)} \frac{1}{\log |deg(a)|}}$$

where $\Gamma(a)$ is the neighborhood of actor a, the set of objects a collaborated on; and , $deg(a) = |\{b \mid (a, b) \in E\}|$ is the total number of objects that are adjacent to actor a.

Now, given G_O, we find clusters $C_1, \ldots, C_m$ of nodes in V_O using the SSDE-Cluster algorithm [Magdon-Ismail and Purnell 2011], which is based on metric embeddings followed by soft Gaussian Mixture Model clustering to find a prespecified number m of overlapping clusters. We choose this algorithm from among those available because it a) provides overlapping clusters; b) is very efficient; the level of cluster overlap can be tweaked easily. One property of SSDE-cluster is that it constructs overlapping clusters which is appropriate since objects could belong to multiple clusters. Categorical clustering algorithms such as [Gibson et al. 2000] could also be used (but these typically give non-overlapping clusters). We mention that there is significant literature in clustering and a comprehensive overview is beyond the scope of this paper.

Each cluster will be treated as a hyperedge by our algorithm, representing a set of objects that are related to each other due to their relationship through common actors. We will use $H(G) = \{C_1, \dots, C_m\}$ to denote the set of hyperedges for the input graph of objects. More details about the clustering algorithm can be found in the Appendix.

Later in the paper, we will investigate the choice of the above distance (weight) function. We will compare this function w_{aa} to the simpler Jaccard function

$$w_{jac}(b_1, b_2) = \frac{|\Gamma_A(b_1) \cup \Gamma_A(b_2)|}{|\Gamma_A(b_1) \cap \Gamma_A(b_2)|}$$

where $\Gamma_A(b) = \{a \mid (a, b) \in E\}$ is the set of neighbors of b of type A.

2.2. Computing Scores

We now get to the meat of the algorithm. We are given a bipartite graph $G = (V, E)$ and a set $H(G)$ of hyperedges of objects in G. The aim is to compute prominence scores v_a, v_b and v_c for each actor, object and hyperedge. Our algorithm is based on the following assumptions:

— Prominent actors contribute to prominent objects.
— Prominent objects tend to be in prominent hyperedges.

However, both relationships tend to be noisy. As a result, we need a robust method to merge values of a set of objects. Suppose X is a set of values. The *top* function aims to improve robustness for this purpose:

$$top(\beta, X) \ = \ \{x \mid x \text{ is in the top } \beta * |V| \text{ highest values in } X\},$$

where $0 \leq \beta \leq 1$; $top(\beta, X)$ contains the top-β fraction of elements in X. Given the noisy set X of values, we are only going to consider the top β percent of the values in determining the quality of the entire set X. As an example, let us consider an author. Suppose X is the set of values for all the papers of the author. If $\beta = 1$, to be a good author all the values in X should be good. However, if we consider someone a good author if their top papers are very good, then we would use $\beta < 1$.

We can also improve the assesment of the set X by also taking into account the worst papers of the author. You might expect that a good author writes a few exceptional papers but no bad ones. For the bulk of the paper we stick to the simpler $top(\beta, X)$ measure; we will explore enhancements later. For now, we will simply fix $\beta = 0.5$.

We now give the iHypR algorithm. It has four basic steps which it repeats in a cycle. First compute the values of the objects based on the values of the authors contributing to each object. Next compute the values of the hyperedges from the values of the objects, and then do the reverse by using the values of the hyperedges to infer values of the objects. Finally, to complete the cycle, use these new values of the objects to recompute values for the actors. The algorithm is summarized in Algorithm 1.

iHypR has similarities to Hits [Kleinberg 1999] and its variations [Borodin et al. 2005]. We highlight the differences: First, iHypR computes scores for three types of nodes: actors, objects and hyperedges. The hyperedges are computed from an object to object graph that is weighted based on the relationships between the actors. Furthermore, it treats actors, objects and hyperedges differently. For actors and objects, it is desirable to have many connections (using Sum like in Hits). For hyperedges, the average value is considered. Finally, the top function introduces robustness while preserving the bias towards bigger sets when necessary.

Algorithm 1 iHypR ($G = (V, E)$)

```
1:  Input: Bi-partite $G = (V, E)$, Compute actor to actor graph $aa$ based on $w_{aa}$, and
    hyperedges $H(G)$ based on $aa$.
2:  For $a \in V_A$, let $v_a = deg(a) / \sum_{a \in V_A} deg(a)$.
3:  **while** not converged **do**
4:      **for all** $b \in B$ **do** $v_b = sum(\{v_a \mid (a, b) \in E\})$
5:      normalize $v_b$ values to sum up to 1
6:      **for all** $C \in H(G)$ **do** $v_C = avg(top(0.5, \{v_b \mid b \in C\}))$
7:      normalize $v_C$ values to sum up to 1
8:      **for all** $b \in B$ **do** $v_b = avg(\{v_c \mid \exists C \in H(G) \,\&\, b \in C\})$
9:      normalize $v_b$ values to sum up to 1
10:     **for all** $a \in A$ **do** $v_a = sum(top(0.5, \{(\frac{v_b}{deg(b)}) \mid (a, b) \in E\}))$
11:     normalize $v_a$ values to sum up to 1
12: **end while**
13: return $\{v_a\}, \{v_b\}, \{v_C\}$
```

Note that the scores of hyperedges in step 6 (similar for authors in step 10) are not a linear function of the input scores due to the top function. As a result, it is not easy to analyze the convergence of the algorithm. However, in practice, the algorithm converges with less than 300 iterations based on all the datasets and settings we have tried. The iHypR algorithm has tunable parameters and its performance can be improved further if additional ground truth data is available. However, the algorithm does not assume the existence of such data or hyperedges. It uses a fixed set of parameters in all datasets we tested. As we will illlustrate in the next sections, the most crucial change that can be made to the algorithm depends on how noisy the underlying data is. In more noisy data sets, a more robust version of the algorithm that does not consider objects is preferred.

Computational Complexity of iHypR. The first step for iHypR computes the object hyperedges. The running time of this step is dependent on the specific clustering algorithm and can be zero if real hyperedges are used. Due to this, we omit it in the overall computational cost. Let $|E|$ be the number of edges in the actor-object-hyperedge tri-partite graph. A single iteration of iHypR takes time $O(|E|)$. The number of edges between actors and objects is specified in the input. The number of edges between the objects and hyperedges depends on the clustering algorithm. Typically an object is in a small (constant) number of hyperedges in which case the number of such object-hyperedge edges is order the number of objects. Since selection is linear, the top function can be computed in linear time as well, and so the running time of the algorithm is essentially linear in the size of the input.

3. EXPERIMENTAL SETUP

3.1. Datasets Used

In our tests, we use three different data sets: DBLP, Enron and IMDB. The details of how these data sets are processed are given in the Appendix. For all these datasets, we have a different external measure of prominence, which is explained below. The details of how these measures are computed are also given in the Appendix.

DBLP. (the Digital Bibliography & Library Project) [3] is a dataset containing information about scientists (actors) from Computer Science, papers (objects) that they

[3]http://www.informatik.uni-trier.de/∼ley/db/

publish and the venues (true hyperedges) that these publications appear in. We use the citations that authors receive for their publications as the external measure of performance. Note that citations are not part of the prominence computation and therefore they are suitable for validating the ranking. It is also widely accepted that citations are a useful measure of prominence. We consider two measures:

— **h:** The H-index value [Hirsch 2005] of an author is h if she has at least h papers with h or more citations.
— **t**: The tc-10 value of an author is the total number of citations of the author's top 10 papers (or all if the author has less than 10 papers).

The H-index incorporates both quality and how prolific an author is, while tc-10 mostly focuses on quality, in particular how well the top papers of the author are viewed in the scientific community.

Enron. email data set [4] contains emails (objects) between employees (actors) of Enron. For prominence, we consider the position (**p**) of the employee in the organization assigned numerical values between 1-7 (1 for CEO and 7 for employee) based on an earlier study [Diesner et al. 2005].

IMDB. (the Internet Movie Database [5]) contains information about movie stars (actors) who star in films (objects). For prominence, we consider the movie budget as a measure of prominence. To overcome problems of inflation and the fact that the movie budget is more like an upfront investment as opposed to a measure of life-long achievement like citations, we consider movie budget info for each decade separately. We first partition the movies into years, then rank them by their budget within the decade it belongs to. Finally, we assign a value to the movie (normalized movie value) by the equation:

$$mv(i) = \frac{k - r(i)}{k - 1}$$

The prominence of the actor in a specific decade is given by the average value (**av**) of her movies given by the above method.

3.2. Performance measures

To measure the performance of an algorithm, we first run it and then rank the actors based on the scores returned. Then, we consider two types of performance measure in this paper:

— **avgx** is the average value of the prominence measure **x** (such as h,t, p or av) for the top 20 actors returned by the studied algorithm. This computes how well the algorithm performs at the top.
— **kx** is the Kendall-tau measure between the ranking returned by the algorithm and the ranking given by the measure **x**. This measures the overall performance of the algorithm.

The *Kendall-tau* measure is given by the number of pairs ordered correctly by the algorithm (as compared to the ground truth ordering) minus the number of pairs ordered incorrectly, all divided by the total number of pairs compared. A Kendall-tau of 1 corresponds to identical orderings whereas -1 corresponds to full reversal. For both measures, higher is better.

[4] www.cs.cmu.edu/∼enron/
[5] www.imdb.com

To compare the performance of a set of algorithms $a_1 \dots a_n$ for a specific performance measure, we first compute the performance $perf(a_i)$ of each algorithm a_i. We then compute the regret of choosing one specific algorithm a_i from the set as the performance loss when choosing a_i versus choosing the best algorithm:

$$regret(a_i) = \frac{(max_{1 \leq k \leq n}\ perf(a_k)) - perf(a_i)}{max_{1 \leq k \leq n}\ perf(a_k)}$$

The best algorithm has regret 0. Algorithms with worse performance have more positive regret values, normalized to a percentage of the max value. The higher the regret, the worse it is to choose this algorithm over the one with the best performance. Note that, as the regret value is relative, when comparing the same result to a different set of results, we may get different regret values.

3.3. Algorithms Studied

We compared iHypR to several well known benchmark algorithms described below. The input to all the algorithms are the actor-object ($A0$) bi-partite graph, the inferred hyperedges (H), and, if available, a proxy for true hyperedges (T) (for example conference venues in DBLP). The output is a ranking of the *actors* (nodes of A).

Our algorithm iHypR can use either the true hyperedges (T) or inferred hyperedges (H). The benchmark algorithms typically do not use hyperedges. In fact, all other algorithms have been designed for some default input that is not AOH. In such cases, these algorithms will ignore some of the input. We will consider some natural extensions of these existing benchmarks to use more of the input than their default, and so there are multiple versions of an algorithm. For example, algorithms may only consider two parts of the input and use $A0$ or AH; or, they may just consider an induced graph on actors A. When just using A, the performance of algorithms may vary depending on how the weights in this graph are computed. We consider 4 approaches to computing such weights:

Name	$w(a_i, a_j)$	
aa (Adamic-Adar)	$\sum_{b \in \Gamma(a_i) \cap \Gamma(a_j)} \frac{1}{\log \lvert\Gamma(b)\rvert}$	symmetric edges
jac (Jaccard)	$\frac{\lvert\Gamma(a_i) \cap \Gamma(a_j)\rvert}{\lvert\Gamma(a_i) \cup \Gamma(a_j)\rvert}$	symmetric edges
log	$\log \lvert\Gamma(a_i) \cap \Gamma(a_j)\rvert$	symmetric edges
$norm$	$\frac{\lvert\Gamma(a_i) \cap \Gamma(a_j)\rvert}{\lvert\Gamma(\Gamma(a_i))\rvert - 1}$	asymmetric edges

where $\Gamma(x)$ is the neighborhood of the set x in $A0$; note that $|\Gamma(\Gamma(a_i))| - 1$ is the number of distinct 'co-actors' of actor a_i.

— *Hits* [Kleinberg 1999] is typically used for directed graphs with a single type of object (in our-case actors). We extend it to a bi-partite graph of two types of objects, t_1, t_2 and compute scores for each type separately. We implement an iterative method that uses the scores of all type t_2 nodes to update the scores of type t_1 nodes and vice versa uxntil convergence. We consider two types of graphs actor-object, actor-hyperedge. So, we tested $Hits_{AO}$, $Hits_{AH}$, $Hits_{AT}$, $Hits_{A_{norm}}$. $Hits_{AO}$ is a version of iHypR that ignores the hyperedges, and we will use it as the benchmark to see what added value can be gained from hyperedges.

Algorithm	avgh	avgt	kth	ktt	avg. regret
iHypR$_{AOH}$	**0.000**	**0.000**	0.005	**0.000**	**0.001**
iHypR$_{AT}$	0.031	0.208	0.124	0.070	0.108
PG_{AOT}	0.163	0.271	**0.000**	0.057	0.123
PG_{AOH}	0.16	0.26	0.01	0.07	0.125
$PG_{A_{norm}}$	0.107	0.093	0.114	0.202	0.129
iHypR$_{AOT}$	0.055	0.326	0.098	0.052	0.133
PG_{AO}	0.24	0.23	0.01	0.06	0.136
$PG_{A_{aa}}$	0.211	0.329	0.045	0.165	0.188
iHypR$_{AH}$	0.37	0.17	0.09	0.17	0.200
$Deg_{A_{norm}}$	0.18	0.11	0.39	0.13	0.201
$PG_{A_{log}}$	0.07	0.51	0.02	0.20	0.203
$InDeg_{A_{norm}}$	0.160	0.373	0.364	0.129	0.256
$Hits_{AO}$	0.725	0.879	0.273	0.311	0.547
$Hits_{AH}$	0.669	0.850	0.227	0.456	0.551
$Hits_{A_{norm}}$	0.855	0.926	0.455	0.456	0.673

Fig. 3. The regret of tested algorithms across multiple performance measures, sorted by average regret. iHypR is clearly the superior algorithm, with essentially zero regret across the board. The true performance numbers corresponding to this table is given in the Appendix in Figure 13.

— PG: ***Pagerank*** [Brin and Page 1998] can be extended to run on directed versions of our graphs. We set α in the pagerank algorithm to $\alpha = 0.85$. We tested PG_{AOH}, PG_{AOT}, PG_{AO}, PG_{AH}, $PG_{A_{aa}}$, $PG_{A_{jac}}$, $PG_{A_{log}}$, $PG_{A_{norm}}$.
— ***iHypR*** I hyper can easily be run AO (only running steps 4,5 and 10 and 11) or on AH (compute steps 6 (substitute a for b),7,10 (substitute c for b) and 11). So we tested iHypR$_{AOH}$, iHypR$_{AOT}$, iHypR$_{AO}$, iHypR$_{AH}$, iHypR$_{AT}$.
— $InDeg$: ***Indegree***, the indegree of an actor in AO (the number of objects the actor collaborated on). Prior work [Borodin et al. 2005] has suggested that indegree is a good substitute for many link based algorithms.
— Deg: ***Degree***, $|\Gamma(\Gamma(a_i))| - 1$, which is the total number of distinct actors a_i has collaborated with.
— *Centrality:* We also tested betweeness and closeness centrality [Faust and Wasserman 1994] using the actor graph A_{jac}. Betweeness computes, for an actor, the number of shortest paths that pass through the given actor, normalized by the total number of nodes. Closeness computes the average distance from an actor to every other node.

4. ALGORITHM PERFORMANCE

Our main experimental test bed is the DBLP data set, because it is huge and we have an accurate external measure of performance based on citation counts.

To measure performance based on citation counts, We consider four measures: *avgh, avgt* are the average h-index and tc-10 values for the top 20 actors, and *kth, ktt* are the Kendall-tau of the same measures for the whole dataset. As mentioned in the previous section, we plot regret, which shows how much worse an algorithm does when compared to best. The best algorithm for a particular measure has regret 0.

Figures 3 and 4 shows the regret of the algorithms we tested, results are ordered by the average regret. Hits overall is not competitive. Clearly iHypR$_{AOH}$ is the top performer. In all except one case, iHypR$_{AOH}$ is the top performing algorithm. Only for kth PG_{AOT} outperforms iHypR *slightly*, by half a percent. Furthermore, there is no other algorithm that performs uniformly well across multiple performance metrics.

We now analyze performance in detail. For avgh, we are interested in authors who have a large number of publications that are well cited. Given that iHypR$_{AT}$ outperforms iHypR$_{AOT}$, it means that the additional paper information seems to increase the

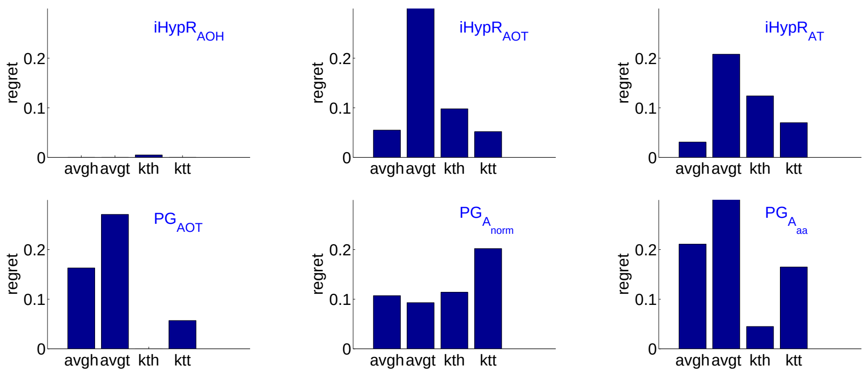

Fig. 4. The regret of top performing algorithms

noise in the data. However, iHypR$_{AOH}$ is not impacted by this and seems to outperform the other versions. We interpret this as follows. True hyperedges are inherently noisy: a conference may have some good and some not so good papers. When we disregard the papers, these noisy values are aggregated at the author level which reduces the noise. When papers are included, the paper value computed from multiple authors counteracts this noise reduction process. However, when we do the same for iHypR, this is no longer a problem. We attribute this to the fact that the *inferred hyperedges* are not as noisy, i.e. inferred hyperedges captures the true correlations between papers and are better at doing so than true edges.

For avgt, iHypR is the best. But, other variations of iHypR perform poorly. The second best algorithm, by a large margin over the others, is $PG_{A_{norm}}$. If we examine $PG_{A_{norm}}$ closely, we see that it pays attention to the social link between the people, but normalized to the energy of each person. All other algorithms perform rather poorly for this measure. This shows that avgh and avgt are fairly different measures: avgh is concerned with “authorities” in the Kleinberg sense, people who are connected to the best venues. However, avgt is more concerned with the individual relationships which impacts their centrality. People who have made a big impact are likely to be central figures in the community, but not necessarily those with large number of papers or strong collaborations. Note that in both cases, our algorithm outperforms both Hits and the Pagerank variations. We again attribute the performance of iHypR to the computation of the hyperedges, which incorporates this notion of community and an author’s place in a community seamlessly.

For kth and ktt, we are interested in the overall performance of the algorithm. The centrality based approach seems to work well for this measure. But, pagerank is not robust across various types of graphs considered. iHypR seems to be best performing for ktt and almost as good as pagerank for kth. We interpret this as that the hyperedges computed by the algorithm contain sufficient information to rank all the authors appropriately.

Based on this, we conclude that iHypR can incorporate a number of important factors in ranking, including a notion of authority and centrality in and across multiple communities. By incorporating these measures, it provides a robust ranking method for social graphs of collaborative activities. No other ranking algorithm is robust across these large number of measures of performance.

4.1. Impact of Hyperedges

Hyperedges vs. No Hyperedges. First we address the question, "Are hyperedges useful?" The answer is yes, and we present the evidence below. For illustration, we compare the performance of algorithms with and without hyperedges. In particular iHypR and $Hits_{AO}$ which is a version of iHypR that does not use hyperedges. We also look at pagerank to conclude that the hyperedges are not only useful for our algorithm but also other standard algorithms. The relevant data from Figure 3 is summarized below,

	avg. regret
iHypR$_{AOH}$	**0.001**
iHypR$_{AOT}$	0.133
$Hits_{AO}$	0.547

	avg. regret
PG_{AOT}	**0.123**
PG_{AOH}	0.125
$PG_{A_{norm}}$	0.129
PG_{AO}	0.547

As can be seen, the addition of hyperedges helps considerably. It is possible to get close to the performance with hyperedges using carefully constructed actor-actor weights in the actor graph, but even still the hyperedges win.

Choice of Hyperedges. Later we will see how the choice of clustering algorithm to obtain the inferred hyperedges impacts the rankings of iHypR. However, with DBLP, we have lots of hyperedge type information: journals, conferences, books, etc. One might argue that a more careful choice of hyperedges from the available ones might help. Indeed this is the case, as indicated by the following table of average regret among the three algorithms below.

	avg. regret
iHypR$_{AOH}$	**0.085**
iHypR$_{AOT:conf}$	0.107
iHypR$_{AOT}$	0.191

In the computer science field the conferences are important and are faithful aggregators of topic and stature. Journals are more "noisy" in that a much wider variety of authors and papers may appear in the same journal. Thus, one might postulate that 'truer' hyperedges are likely to result by ignoring the hyperedges defined by journal venues. This is what the $T : conf$ are. However, we observe that i) inferred hyperedges are still better than carefully chosen true hyperedges; ii) considerable insight into the data is needed to obtain such true hyperedges, and such information would not be available in most data sets. To some extent, these results are also revealing about the nature of DBLP, in that our initial hypothesis seems to be correct: conferences *are* the more reliable source of hyperedges.

To summarize, hyperedges are important; and, if using true hyperedges, the choice ot the hyperedges needs some thought. However, inferred hyperedges require no thought and perform better!

4.2. Actor hyperedges vs. Object hyperedges

In this section, we consider whether actor hyperedges found by clustering actors adds any value. We consider the graph A_{aa} cluster actors using the same algorithm and test three versions of our algorithm: (1) iHypR$_{AOH}$ is the regular iHypR algorithm, (2) iHypR$_{HAO}$ considers actor hyperedges, actors and objects, and (3) iHypR$_{HAO}$ considers actor hyperedges, actors, objects and object hyperedges. Note that the algorithm iterates from left to right and right to left over the input types. When considering actor hyperedges, we use the same aggregation method as object hyperedges. The results given in Figure 5 show that object hyperedges are much more useful in ranking than actor hyperedges, even though both types of hyperedges are overlapping. One possible

Algorithm	avgh	avgt	kth	ktt
SSDE	**0.00**	**0.00**	**0.00**	**0.00**
RC20	0.79	0.92	0.09	0.11
RC50	0.77	0.89	0.14	0.18
FC	0.61	0.53	0.11	0.11

Fig. 6. Impact of clustering algorithm on iHypR performance: regret of using different clustering algorithms

explanation for this is that people may belong to many different groups of collaborations. But even overlapping clusters tend to put people in boxes, groups that they best fit. Given an actor may have many objects, one can capture this multi-faceted nature of actor's interest by finding common groups of objects.

Algorithm	avgh	avgt	kth	ktt
iHypR$_{AOH}$	**0**	**0**	**0**	**0**
iHypR$_{HAOH}$	0.04	0.06	**0**	0.04
iHypR$_{HAO}$	0.17	0.26	0.02	0.07

Fig. 5. Regret of using actor and object hyperedges in the DBLP dataset.

4.3. Impact of clustering algorithm choice

The above results use the SSDE-clustering algorithm [Magdon-Ismail and Purnell 2011] by default. The algorithm determines the number of clusters using a specific methodology described in the Appendix. In particular, we find first a set of top clusters roughly corresponding to research areas and then cluster within each cluster which finds the research collaborations within each research area. SSDE finds overlapping clusters. The total clusters used in the case of DBLP was 100.

We may also use hyperedges found by different clustering algorithms for comparison. Again, we find that the choice of clustering algorithm matters, consistent with the findings of the previous section that the hyperedges are important. FastCommunity [Clauset et al. 2004] (FC) finds densely connected subgroups of the graph and produces disjoint clusters of varying sizes and number depending on the dataset. We used the adamic-adar weights to define the graph on the objects for the clustering algorithm. Total number of clusters is 2638.

RankClus [Sun et al. 2009a] takes a dataset with predefined venues and partitions these venues into n disjoint clusters where n is an input to the algorithm. The idea is to cluster the venues into specific research areas. There is no specific method given to find the total number of clusters, so we report on two runs with 20 and 50 clusters. Other details for these algorithms are given in the appendix.

Figure 6 shows the regret of choosing different clustering algorithms. SSDE outperforms all the others in this setup (no surprise, since that is the one we *chose* to recommend as the default), but what the result indicates is that the choice of clustering algorithm should obtain good (possibly overlapping) clusters. In particular, FC performs reasonably well overall. For this reason, we will investigate FC in more detail below. We also observe that a clustering methodology that uses venues is not a good idea, because it gives up some of the benefit of inferring the hyperedges by being tied down to the venues.

4.4. Impact of the distance measure when clustering

Recall that we introduced two distance measures in Section 2.1. Each is considering how important the link between two papers is. The first function w_{aa} (Adamic-Adar), the default, is based on an interpretation of the Adamic-Adar measure. It looks at all

Algorithm	Dist. F.	avgh	avgt	kth	ktt	avg regret
SSDE	w_{aa}	0.02	0.02	**0.00**	**0.00**	**0.01**
SSDE	w_{jac}	0.19	**0.00**	0.07	0.04	0.07
FC	w_{aa}	0.61	0.53	0.12	0.11	0.34
FC	w_{jac}	**0.00**	0.20	0.12	0.07	0.10

Fig. 7. Impact of different distance measures in the performance of the algorithm, the regret value for different algorithms. SSDE is using 10.10 clusters in both cases

common authors between the papers and considers the collaboration significant if the authors do not have many other papers. The function w_{jac} on the other hand does not take this into account. The more common authors two papers have, the more related they are. In this section, we will consider the importance of the distance function in algorithm performance. We will investigate two clustering algorithms SSDE and FC using both types of functions.

Figure 7 shows the impact of the different distance measures. SSDE performs better than FC overall for both functions. However, w_{jac} clearly outperforms w_{aa} for FC, but not for SSDE. One of the reasons for this is the fact that FC optimizes a local criteria, density of connections, while SSDE optimizes a more global criteria, i.e. distances between nodes. The distances produced by the function w_{aa} are more meaningful to a clustering algorithm that uses a global criteria to interpret the meaning of this distance. For example, the fact that a person has many papers becomes meaningful when compared to the rest of the graph, how many papers other people have. This criteria is what makes w_{aa} different. However, the function w_{jac} is based on individual papers and is a more local criteria. Hence, it is more appropriate to use in an algorithm that has a more localized view of the meaning of distance, such as FC, which is based on the density of the cluster's neighborhood. But, the value of the edge distance is not interpreted in relation to the edge distances in the whole dataset.

One expects that a clustering algorithm with a global criteria for interpreting distances will perform best for the iHypR algorithm if: (1) the distances are meaningful for prominence, in the case of DBLP, and (2) there are big differences for the distance in the dataset.

4.5. Exploiting community substructure

In this section, we investigate the difference of clustering scheme in the performance of the algorithm. We compare clustering the dataset to top 100 clusters vs. clustering it to 10 top clusters first, and then clustering the objects in each cluster to 10 more smaller clusters (10.10, still producing 100 clusters). So far in our algorithm, we have been using the 10.10 scheme with w_{aa} as the default method. The regret value for using each clustering scheme for different distance measures for SSDE are shown in Figure 8. The results show that considering subclusters is beneficial in w_{aa}, but not in w_{jac}. Consistent with results from 4.4, the distances produced by w_{aa} are relative to the whole graph. But, there is a great variation in these values across communities. For example, it is natural for a strong theoretician to produce only a few impactful papers, while a strong systems person is likely to produce a larger number of papers to make impact. Hence, when using w_{aa}, it is clearly beneficial to use subclusters.

When using w_{jac}, this type of distinction is not clear as we only consider the pairwise similarities of papers based on their common authors. So, this distance measure is agnostic to this type of distinction between communities. Two papers in systems and theory will exhibit the same type of distance and papers across areas are less likely to be connected in the graph. In this case, by using sublusters (10.10), we loose an

Dist. F.	Cluster Size	avgh	avgt	kth	ktt	avg regret
w_{jac}	10.10	0.17	0.11	0.07	0.04	0.10
w_{jac}	100	0.07	**0.00**	0.11	0.04	0.05
w_{aa}	10.10	**0.00**	0.13	**0.00**	**0.00**	**0.03**
w_{aa}	100	0.36	0.70	**0.00**	0.04	0.27

Fig. 8. Impact of clustering the whole dataset vs. clustering each community separately using SSDE, regret of different cases.

Case	avgh	avgt	kth	ktt
a priori best	**0**	**0**	**0**	**0**
iHypR	0.03	0.36	0.02	**0**
Use average in step 4	0.74	0.91	0.14	0.22
Change $v_b/deg(b)$ in step 10 to v_b	0.09	0.49	0.02	0.04
median instead of avd in step 10	0.03	0.36	0.02	**0**
median instead of avg in step 6	0.02	0.30	**0**	**0**

Fig. 9. The regret of changing various algorithm settings, compare values to the maximum for each specific performance measure.

opportunity to find big outliers across the whole data set. We see that w_{jac} with 100 clusters captures this result. This case has the highest $avgt$ value.

The conclusion from this study is then as follows. If a dataset has subcommunities with different characteristics, then:

— It makes sense to use a clustering algorithm with a global method for interpreting distances and w_{aa} with subclusters when optimizing for Kendall-tau values, and to use w_{jac} without subclusters to optimize for the top values returned.
— However, localized clustering algorithms will also perform well. However, for these w_{jac} is a better choice.

4.6. Variability across different parameters of the iHypR algorithm

The iHypR algorithm uses different algorithms at each step, sum of all values for step 4, average of top 50% values at step 6, average of all values in step 8 and sum of the top 50% values in step 10. The final question we would like to answer is how sensitive the algorithm is to the choice of these functions. First of all, we review the reasoning behind these functions.

At step 4, we compute the value of objects as a function of the actors in it. In general, the more people are associated with an object, the more valuable it is. This is based on the assumption that larger number of actors contribute to higher quality in general. We also try the average value in step 4 as an alternative.

At step 6, we are computing the value of hyperedges as a function of the objects in it. In this case, collaboration size is not significant. Instead of sum, we use average. As an example, consider a conference. A large conference is not necessarily a better conference. In our case, we do not penalize a venue for its size which would hurt performance considerably. However, we vary the input to the top function as discussed below.

At step 8, we do not penalize or favor an object for belonging to multiple hyperedges. Hence, we simply use average. For example, a multi-disciplinary research paper is not more or less prominent compared to the rest.

Finally, at step 10, we tie the prominence of the actor to the value of his/her objects as well as their number. The higher the number of such objects, the more prominent the author. In fact, using average in this case would also hurt performance considerably and therefore we do not consider it. However, we consider whether to use $v_b/deg(b)$

α	β	avgh	avgt	kth	ktt
0.85	0.1	**0**	**0**	0.09	0.07
0.85	0.5	0.02	0.32	0.00	0.04
1	0.5	0.03	0.32	**0**	**0**
1	1	0.03	0.33	0.02	0.04

Fig. 10. The regret of changing settings for the top function in steps 6 and 10. iHypR corresponds to $\alpha = 1, \beta = 0.5$.

which assigns value to the object proportional to the number of actors associated with it, or v_b which disregards this issue. We also consider different top functions for this step.

For steps 6 and 10, we consider using the top function with β = 0.1, 0.5 or 1, i.e. consider 10%, 50% or all of the objects. We also consider a variation of the top function which also considers the value of bottom objects using $\alpha * t + (1 - \alpha) * b$ where t and b are the sum/avg of the value of top and bottom k% of objects and $\alpha = 0.85, 1$. This gives us the 4 cases shown in in Figure 10. We use the same setting for all top instances in the algorithm, though given each paper has only 2-3 authors, it generally only makes a difference for hyperedges.

We compare the performance of the algorithm for these cases to an idealized case, in which the performance is maximized for a specific measure. We pick the max for each metric over the 40 cases we tried and show the regret against this value. Note that in many cases, the setup that achieves the top avgh may be different than avgt. The results are given in Figure 9. We see that the algorithm is not very sensitive to changes in various parameters with the exception of step 4. For this dataset, it is crucial to take into account that higher number of collaborations means better objects. A similar trend is followed in the Enron dataset which we will examine in the next section: the higher number of people who are privy to some information, the more valuable it is. However, this is only true for information that is not broadcast to everyone. This is something we pay attention to in data cleaning. For IMDB, we only look at the top k actors, so this is not immediately applicable.

We also see that the most sensitive performance measure is avgt. Hence, if this measure is of concern, then one can significantly improve performance over iHypR by tuning the algorithm to a specific case. To find researchers with significantly high citations, we need to consider top and bottom 10% at step 10 and also use 100% at step 6. In other words, consider hyperedges uniformly good and judge authors for their very top papers. However, we also note that the avgt measure is a very sensitive measure, only looking at the top 20 in the ranking. One should be careful in paying too much attention to this measure. One way to make it more stable is to increase 20 to say 100, or 1000, or 1%. For illustration in this paper we took 20, because it is a reasonable question to ask “Who are the top 20 actors?”, but naturally the answer will be very sensitive to precisely how you define “top” since we are only considering 20 out of about a million.

5. OTHER DATASETS

Based on our analysis of the DBLP dataset, we now consider two other datasets and analyze how well iHypR performs for these datasets.

5.1. iHypR with Enron Dataset

To illustrate the performance of our algorithm, we use the Enron email dataset. Details of the dataset and our processing of it is given in Section 3. We show the average rank of employees returned by the algorithm for each position: from 1 as best (CEO,

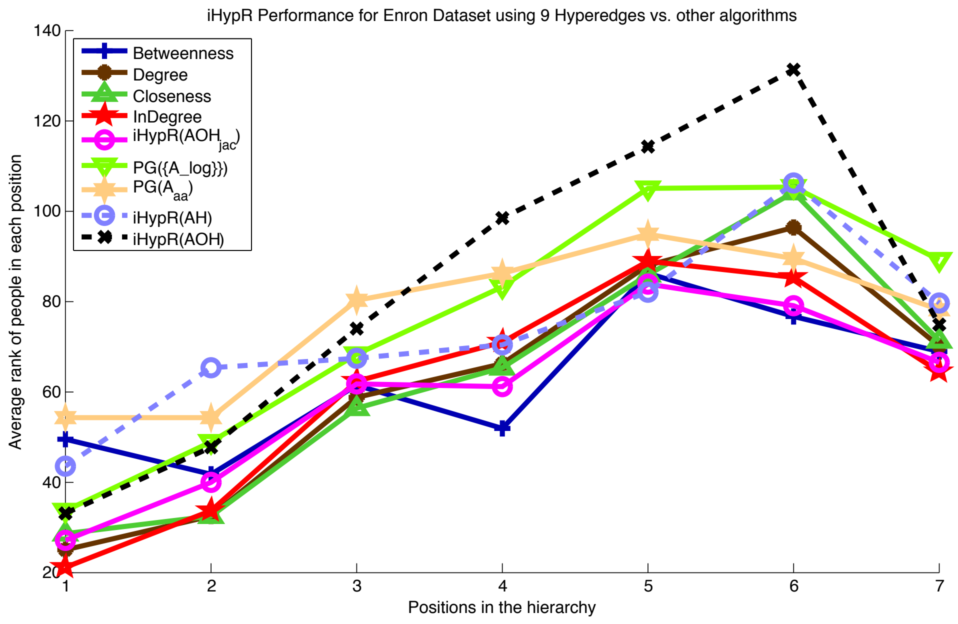

Fig. 11. Average rank of employees in the Enron dataset (Note: Betweenness, Degree, Closeness and Indegree are using graph AA/J).

President) to 7 (Employee) for various centrality measures and our algorithm using 9 clusters in Figure 11. We have also shown the Kendall-tau of rankings given by the hierarchy and those provided by our algorithm in Figure 12. Due to the high number of ties, when computing the Kendall-tau measure we only look at those cases where the hierarchy poses an ordering and the studied algorithm reverses the ordering. The hierarchy of the positions is taken from [Diesner et al. 2005]. We employ a directed interpretation of our algorithm: mails are as valuable as their sender, and people are as valuable as the mails that they receive. We test iHypR using various graphs: AH and AOH using w_{aa} and AOH/J using w_{jac}.

Table I. Number of employees in our dataset for each position of the Enron hierarchy.

Position	Number of Employees	Position Description
1	8	President, CEO
2	28	Vice President, Managing Director, Director
3	16	In House Lawyer
4	22	Manager
5	16	Trader
6	4	Analyst
7	24	Employee

One limitation of this experiment is that the relative importance of the positions in the hierarchy are not known. But we expect the ranks to go up as we go down in the hierarchy: an analyst (position 6) should be lower ranked compared to a manager (position 4). But it is not necessary that the change is a linear line or that all flips in hierarchy should have the same penalty. As a result, both the average rank or the Kendall-tau measures are imperfect in reflecting the true performance.

First, looking at the graph in Figure 11, we expect the average rank to increase for higher positions in the hierarchy. However, almost all algorithms do badly for position

Algorithm	Kendall-tau	Regret
iHypR_{AH}	**0.04**	**0**
$PG_{A_{aa}}$	0	1.05
$Betweenness$	-0.02	1.56
$\mathsf{iHypR}_{AOH_{jac}}$	-0.04	1.92
iHypR_{AOH}	-0.10	3.49
$InDegree$	-0.13	4.21
$PG_{A_{log}}$	-0.13	4.31
$Degree$	-0.17	5.41
$Closeness$	-0.19	5.82

Fig. 12. The kendall-tau and the regret of kendall-tau for the ranks in the Enron data sets for the tested algorithms.

7, i.e. the employee level. One reason for this could be that some employees are assistants of the higher level employees and most e-mail to the people in these higher positions are routed through them. As a result, the importance of these employees reflect the importance of their immediate boss.

If we consider positions 1-6, we see that iHypR seems to have the steepest change in terms of ranks which seems to indicate that the rank given by iHypR correlates best with the position in the hierarchy. However, this figure does not provide a good illustration of the performance of the algorithm for pairwise ranking of people. By considering the Kendall-tau measure in Figure 12, we note that iHypR_{AH} is by far the winner in this case, having the only positive correlation. This could be attributed to the nature of this dataset as discussed in the introduction.

Hyperedges provide additional information about relationships between people, improving performance over all algorithms that consider only the relationships between actors. However, in this network, emails are noisy indicators of relationships between people. Especially, those who are carbon-copied in emails are only tangentially related to the senders. As a result, using emails in the algorithm (i.e. iHypR_{AOH}) improves the overall ranking of nodes, but does not help in pairwise ranking. The more robust version of our algorithm iHypR_{AH} is the winner in this case. Similarly, for iHypR, when considering the Kendall-tau measure, w_{jac} outperforms w_{aa} (-0.04 vs. -0.1). The results remain the same when we consider 9 top clusters or 3 top and 3 secondary level clusters for both measures. This could also be due to the noisy nature of emails, the neighborhood information used in w_{aa} is much more noisy than in DBLP. Writing papers together is a much more selective process than including someone in the cc of an email.

5.2. iHypR with the IMDB Dataset

Next, we analyze the IMDB dataset using iHypR. As mentioned in Section 7, we have divided up IMDB into decades and look at the average performance of actors in each decade. We find that there are about 16 clusters per decade. One thing we noticed for IMDB that it does not seem to benefit from subclusters and in fact there is a slight performance penalty. One reason for this is that our clustering algorithm is not able to identify the different groups in this dataset such as adult or Asian movies very clearly when multi-level clustering is used. As a result, we use the top 16 clusters in our tests. In these tests, we use closeness centrality.

We first give results based on actor values (*av*) computed based on movie budgets as described in the appendix for each each decade. We consider two measures: *avgav* and *ktav*, the average actor value for top 20 actors and the Kendall-tau of the ranking of all actors returned by the algorithm. The regret of *avgav* is given in Table II. We see that iHypR performs poorly in this dataset, but iHypR$_{AH}$ is the top performing algorithm, followed by degree. However, pagerank variations as well as other centrality measures do not perform as well. Similarly, for *ktav* given in Table III, iHypR$_{AH}$ is the top performer by a wide margin. It seems in this dataset, the movies are too noisy indicators of an actor's prominence. However, the hyperedges of movies containing similar actors tend to have actors of similar prominence, and hence improves the result considerably. An explanation for this is that in iHypR, we use actors to determine the prominence of movies. However, there are many other additional factors that may determine the value of a movie: the writer, producer, cinematographer, production and distribution company.

Table II. Regret of avgav for different algorithms in the IMDB dataset

REGRET	1930s	1940s	1950s	1960s	1970s	1980s	1990s	2000s	Avg
iHypR$_{AOH}$	0.20	0.61	0.39	0.25	0.08	0.73	0.61	0.03	0.41
iHypR$_{AH}$	0.16	0.31	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.02	**0.07**
$InDegree$	0.46	0.52	0.28	0.09	0.19	0.17	0.22	0.11	0.28
$Degree$	**0.00**	0.22	0.02	0.10	0.13	0.21	0.10	0.03	0.11
$Centrality$	0.22	**0.00**	0.30	0.39	0.21	0.35	0.09	**0.00**	0.22
$PG_{A_{log}}$	0.51	0.58	0.27	0.33	0.28	0.26	0.31	0.13	0.36
PG_{AO}	0.33	0.52	0.39	0.23	0.33	0.28	0.33	0.28	0.35

Table III. Regret of ktav for different algorithms in the IMDB dataset

Algorithm	1930s	1940s	1950s	1960s	1970s	1980s	1990s	2000s	Avg
iHypR$_{AOH}$	0.71	1.28	0.73	0.51	0.18	1.35	0.47	**0.00**	0.75
iHypR$_{AH}$	0.23	0.35	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.03	**0.08**
$InDegree$	0.23	0.64	0.19	0.37	0.48	0.50	0.46	0.74	0.41
$Degree$	**0.00**	0.18	**0.00**	0.33	0.44	0.47	0.37	0.59	0.26
$Centrality$	0.60	**0.00**	0.61	0.48	0.36	0.40	0.44	**0.00**	0.41
$PG_{A_{log}}$	0.41	0.67	0.43	0.64	0.68	0.73	0.68	0.93	0.60
PG_{AO}	0.36	0.83	0.38	0.57	0.66	0.71	0.72	1.00	0.61

Note that these findings are based on the movie industry view of the prominence of the actors, not the outside view. Our method is able to find the power circles in each decade that actors belong to by clustering the movie to movie graph. We ask the question what happens if we used movie gross instead of movie budget to determine prominence? In movie gross, the prominence of individuals is determined by the actual success of the movie. However, the movie gross information is known to be very noisy. The interesting thing we find is that in the case of ktav, iHypR$_{AH}$ is still the top performing function overall (avg. regret: 0.21) and iHypR (avg. regret: 0.51) is highly superior to pagerank (avg. regret: 0.94). As for avgav, centrality becomes more and more relevant in 2000s and is the top performer (avg. regret: 0.08) followed by iHypR$_{AH}$ (avg. regret: 0.20). Hence, the communities the actor's belong to seem to play a role in the overall prominence of the actors for external viewers as well. This is not a surprising result as movies with such actors tend to get higher budgets which implies a higher level of publicity. Overall, iHypR$_{AH}$ tends to perform well in cases where the prominence is predominantly determined by the social structure.

Clearly, many different measures of prominence can be considered for movies. This is due to the subjectivity of what constitutes a good movie. So far, we have seen movie budget and gross. How about the overall ratings of the movies by viewers? While budget and gross are highly dependent on the level of investment the studios made to the movie, the ratings are a measure of quality. Unfortunately, there are many different ratings possible for a movie. We will use the IMDB ratings to illustrate the performance of the different algorithms. The prominence of an actor is given by the average rating of their movies in a decade, denoted by *rt*. We compute, *avg-rt* for the average rating of the top 20 actors returned by the algorithm and *kt-rt* for the Kendall-tau measure of the rating values of actors. The results are shown in Figures IV and V. We see a similar trend: iHypR$_{AH}$ is the top performing algorithm by far, outperforming iHypR and many other algorithms considerably.

In summary, there are many outside criteria for prominence of actors: based on the amount of money the industry invests in them, the amount of money their movies bring or the overall (subjective) quality of their movies. For all of these criteria, iHypR$_{AH}$ is the top performing algorithm by taking into account the social component: people who appear in top tier movies tend to mostly appear in movies of similar caliber. Considering movies additionally does not improve performance because there are many additional outside factors that can impact the prominance of movies for every one of the different criteria we considered. But, the input to the algorithm only contains the top three actors and the director of the movie and as a result results in a very noisy indicator of the prominence of the movies.

Table IV. Regret of avg-rt based on average movie ratings for different algorithms in the IMDB dataset

REGRET	1930s	1940s	1950s	1960s	1970s	1980s	1990s	2000s	Avg
iHypR$_{AOH}$	0.05	**0**	0.04	0.08	0.11	**0**	**0**	0.07	0.04
iHypR$_{AH}$	0.03	0.06	0.07	**0**	**0**	0.02	0.02	**0**	**0.02**
$Indegree$	0.06	0.03	0.03	0.10	0.09	0.14	0.18	0.03	0.08
$Degree$	**0**	0.05	0.10	0.01	0.04	0.07	0.15	0.03	0.06
$Centrality$	0.07	0.03	0.14	0.15	0.10	0.31	0.14	0.12	0.13
$PG_{A_{log}}$	0.04	0.05	0.13	0.08	0.09	0.20	0.30	0.11	0.13
PG_{AO}	0.07	0.02	**0**	0.10	0.08	0.16	0.21	0.06	0.09

Table V. Regret of kt-rt based on average movie ratings for different algorithms in the IMDB dataset

Algorithm	1930s	1940s	1950s	1960s	1970s	1980s	1990s	2000s	Avg
iHypR$_{AOH}$	0.62	1.15	0.15	0.58	0.11	1.80	**0**	2.33	0.84
iHypR$_{AH}$	0.23	**0**	**0**	**0**	**0**	**0**	0.19	1.21	**0.20**
$Indegree$	0.14	0.15	0.10	0.62	0.60	0.62	0.86	1.00	0.51
$Degree$	**0**	0.08	0.26	0.66	0.71	0.98	1.01	1.88	0.70
$Centrality$	1.11	0.23	1.69	0.67	0.79	1.04	0.90	4.28	1.34
$PG_{A_{log}}$	0.29	0.50	0.67	0.99	0.98	0.90	0.95	0.89	0.77
PG_{AO}	0.25	0.35	0.38	0.81	0.82	0.74	0.86	**0**	0.53

6. RELATED WORK

In ranking people for prominence, measures based on centrality and node degree can frequently capture the importance of people in a social network [Faust and Wasserman 1994]. For example, degree centrality, a visibility metric of an actor, measures the number of neighbors of an actor. *Indegree* is a special case of degree centrality which measures the number of in-coming links of an actor. This method had been used in early web search engines. The idea behind indegree is that a prominent actor should be pointed to by many other actors in the network. However, higher indegree value does not directly imply prominence of an actor (consider an actor with very high indegree all of whose neighbors are isolated from the rest of the network). The actor has higher visibility in term of indegree score, but he is not prominent at all as he is located in an isolated component of the whole network. Also, indegree treats all incoming links equally. However, actors who are prominently pointed to by many other important actors should be more important than those who are pointed to by other ordinary actors. Closeness centrality measures the distance from one node to all the other nodes in network. In a way, the lower the closeness centrality of a node, the quicker information can travel from any node to this node. These methods have been applied to analysis of scientific literature [Xiaoming Liu and de Sompel 2005] by considering the co-authorship relationship between people, disregarding the actual papers written and the venues they appeared in.

Overall, the performance of these methods in modeling prominence of nodes depends on the underlying goal of the network and the special advantages the links provide to the nodes. While these methods are robust across many different graphs of social networks, they are typically designed and tested on individual communities and will typically not perform well when a large network has multiple communities - hence the need for algorithms that take into account some form the community structure such as [Sun et al. 2009b].

These social network analysis methods have been further adopted for the Web and designed to work for very large graphs [Brin and Page 1998] and the notion of expertise on a specific topic [Kleinberg 1999]. In the Web graph, pages are treated to represent entities, similar to people, and the links between them to represent an endorsement or citation. The eigenvector centrality based methods (Pagerank algorithm [Brin and Page 1998] and its variations [Lempel and Moran 2001; Borodin et al. 2005]) are shown to work well on the Web graph. Lempel and Moran [Lempel and Moran 2001] devised an approach to combine the PageRank and HITS algorithms by performing a random walk on the hub-authority graph and alternating between these the hubs and authorities. They also identified tightly knit community (TKC) effect which dramatically impacts performance of the HITS algorithm. The TKC is a small set of sites in which each site connects to almost all of the others in the set. With this in mind, Borodin et al. [Borodin et al. 2005] developed HubAvg, AT(k), and Max family algorithms based on the HITS to overcome TKC effect. Note that our algorithm improves on the TKC effect in multiple ways. First, the top function limits the influence of a specific set of nodes by considering only a percentage of the values. This way, a larger set with some lesser quality objects can still compete with a smaller and tightly connected set. We also use average instead of sum for larger sets. Also, as the hyperedges are found by clustering algorithms, in some cases reducing the number from thousands to a hundred and allowing overlaps, dense subgraphs are much more likely to embedded in larger and less dense graphs. This may also help reduce the impact of TKC in the overall ranking.

Xi et al. [Xi et al. 2004] propose a generalized model – “link fusion”, which is a unified link analysis framework. This framework considers both the inter- and intra- type link structure among multi-type inter-related data objects. It is shown that the PageRank and HITS algorithms are special cases of the unified link analysis framework. This work does not take into hyperedges of objects and how algorithms can be customized to networks of social collaborations, which is what we concentrate on.

Similar to the earlier centrality measures, pagerank and Hits variations consider graphs containing a single type of node, whereas we only consider three types of nodes. However, Hits [Kleinberg 1999] introduces a bit more structure as it computes two types of prominence: for expertise (authority) and for informativeness (hubs). Given that expertise for a large graph is not meaningful as there may be many topics present, the algorithm first finds a subgraph of the Web that is focused on a specific topic, and then applies the algorithm to this graph. Hence, it integrates a second type of information, a “latent topic” on top of the graph. Note that, our algorithm can be considered a version of Hits if we replace the top function with “sum”. However, our algorithm computes three types of prominence, each node having a single type of prominence measure. In constract, Hits has two types of prominence for each node. Furthermore, we show that object hyperedges are crucial to the performance of prominence algorithms in social networks and how to compute these hyperedges from a given network of collaborations.

Various algorithms have been developed to impose and utilize from an additional structure on top of the Web graph. Haveliwala [Haveliwala 2002] proposed a topic-sensitive PageRank by using a set of representative topics to compute a set of topic specific PageRank vectors instead of a single PageRank vector as in the PageRank algorithm. Using the biased jump probabilities introduced in the original pagerank paper, each PageRank vector is biased with the corresponding topic to improve rank scores of Web pages in the topic. Gyöngyi et al. [Gyongyi et al. 2004] implemented a PageRank-like algorithm, TrustRank, to overcome Web spam by first assigning "trust" to a subset of the nodes with the help of human experts or other methods. The algorithm then takes the information about trusted nodes and propagates trust values to other Web pages based on the link structure. Despite these additions, the algorithms do not make a distinction between all three types of nodes and the relationships that we consider: entities, objects they own and the classes the objects belong to (like topics). We show that by making use of all three, we are able to get a much better prominence ranking across a large range of social network graphs.

Recently, a number of algorithms have been proposed to make use of multiple types of links for different graphs. PopRank [Nie et al. 2005] is an extension of pagerank that treats Web objects with different roles resulting in a heterogeneous network. This algorithm first asks field experts to give a partial rank of objects. The algorithm is then trained by the partial rank to find the appropriate propagation factors between different types of objects. Objects are then ranked using these propagation factors. Balmin et al. [Balmin et al. 2004] proposed ObjectRank which takes heterogeneous information networks as input. This algorithm applies a PageRank-like random walk on the network containing multiple-type objects, while a random-jump action is applied to inter-type components with a specific transition schema between them. This schema is an input to the algorithm and specifies how much authority is transferred along different edges. However, the transition parameters are finely tuned to a subset of objects in the DBLP dataset, requires training data and existing hyerpedges. Our algorithm on the other hand only requires author and object networks, does not require any additional training data.

In [Jeh and Widom 2002], Jeh and Widom devised SimRank to compute similarity of pairs of objects based on the assumption that similar objects are related to similar objects using a extension of the HITS algorithm. This notion of similarity is captured in our clustering algorithm for objects. Similar to the spirit of this method, our clusters are overlapping, allowing an object to belong to multiple clusters of similar objects. In the same vein [Bao et al. 2007], Bao et al. introduced social annotations into the ranking algorithm by arguing that social annotations improve Web search quality. Two algorithms were proposed: Social similarity rank (SSR) calculates the similarity between social annotations based on common pages that use them, then uses this similarity to find similarity between a web query and a web page; Social PageRank (SPR) captures the popularity of web pages by propagating popularity from pages to users who annotate them, from users to their annotations, from annotations to web pages, from web pages to annotations, annotations to users and finally from users to pages again. A similar idea was also implemented in [Hotho et al. 2006]. In essence, this method considers three types of nodes and passes scores between all possible pairs of node types in a single iteration. Our algorithm in contrast works in a more structured setting: actors derive prominence from their objects and objects derive prominence from the hyperedges they belong to. This is based on the assumption that collaborating on an object implies a social tie between the actors. However, there is no such tie between two users who tag the same page or use the same annotation. As a result, our algorithm is not directly applicable to this dataset.

In [Sun et al. 2009a], Sun et al. argues that ranking actors globally in large networks is not meaningful. Actors have prominence in specific communities, which can be found using a clustering algorithm. NetClus [Sun et al. 2009b] extends this idea to three-types networks with star schema. These algorithms take as input the number of clusters, which is assumed to correspond to the number of communities in the dataset. The clustering algorithm is tied to a HITS based algorithm for scoring nodes, where hyperedges with similar actor scores are grouped together in the same cluster. This approach assumes that hyperedges already exist in the data. In contrast, we show with experiments that even if true hyperedges are known, they may sometimes be noisy. Hence, we must first find the correct hyperedges for the objects. Our algorithm in fact first finds the correct hyperedges instead of grouping existing hyperedges. Our hyperedges can overlap, while in [Sun et al. 2009a], they have to be disjoint. Furthermore, we use these hyperpedges to infer the global ranking of the actors instead of each community separately. In our experimental results, we report on tests using the hyperedges found by this method.

Another common difference between our method and various others in the literature is that we report on tests using multiple data sets and multiple performance measures against and external ground truth. This provides insight on the strengths and weaknesses of our method and the types of data that are suitable for our algorithm.

There are other methods that operate on similar hypotheses to ours. For example, collaborative filtering tries to find similar objects based on the similarity of people interested in them. However, there is no social link provided by people's interests. Similarly, many systems that allow users to rate information use a way for people to rank each other. These rankings determine the reputation of a poster. In fact, if the votes and voters are known, then a social tie exists between the users [Adamic et al. 2011]. In such cases, our algorithm can be applied to improve ranking.

There is research that supports the hypotheses used in our work. For example, in social networks, high degree nodes tend to associate with other high degree nodes where degree constitutes a notion of prominence [Newman 2003]. This general tendency of associating with similar others, i.e. assortativity have been illustrated in sociological literature [Rivera et al. 2010]. Research shows [Jones et al. 2008; Wuchty et al. 2007] that especially in scientific research, there is a growing emphasis towards team based work. Team based papers get cited more and teams including top-tier universities produce higher impact papers. As a result, the hypotheses of our algorithm are likely to hold for a data set of research papers. In communication networks, email exchanges do not typically have inherent value but are indicative of underlying organizational structure [Diesner et al. 2005]. Our experiments show that there is still considerable gain in performance when using object and hyperedge information.

This work extends our previous work [Adalı et al. 2011] which introduced an earlier version of the iHypR algorithm with various tunable parameters and showed that it has superior performance to indegree and pagerank for DBLP. The current paper extends the algorithm by removing all tunable parameters and introduces many new results for the DBLP dataset and adds many other algorithms, different weighting methods for object graphs and two new datasets for testing.

7. CONCLUSIONS AND FUTURE WORK

We developed a novel algorithm for ranking in a social network of collaborations where the "semantics of the ties" can add significant value. The main message is that when the social tie between actors is infered by their collaboration in some object, the properties and relations between those objects can significantly improve the ranking (as opposed to only using the social ties among the actors).

In both Enron and DBLP, our algorithm provides significant improvement over a number of other algorithms. In both cases, the emails and papers are based partially on self selection. They can be organized into groups that convey important information about collaboration groups. In IMDB where the casting decisions are based on the perceived marketability of the actors, the movies themselves have little value in the ranking. The social links are what counts. In this case, a different version of our algorithm is the winner.

One important aspect of our algorithm is that it does not need to know how the objects are organized into groups, because the hyperedges groups can be inferred. In fact performance was enhanced by the more robust organization of the artifacts deduced using the overlapping clustering algorithm in [Magdon-Ismail and Purnell 2011].

Our results indicate that methods based on the actor-object-hyperedge graphs have significant potential, and there are several avenues for further investigation. Are there specific properties of the network that can be measured to indicate whether it exhibits subcommunity structure? Can we tune how the information from each node type is incorporated into the ranking for a specific dataset? Should all the hyperedges be used in the algorithm? One might conjecture that only the “important” groups should be used, which is similar to the notion of trust-rank, where the structure of the graph with respect to important nodes can be more robust [Gyongyi et al. 2004]. Another interesting extension is to consider the impact of the sequence of iteration in the quality of the output. Currently, actors derive value from their objects and objects and hyperedges derive value from each other. However, it is possible to consider other sequences of iterations, where authors derive value directly from hyperedges. Another possibility to consider whether authors belong to specific hyperedges that correspond to their “peers”, and whether belonging to such hyperedges brings additional value to the ranking. It would be interesting to study whether such extensions improve the ranking and/or the robustness of the algorithm.

Appendix

In this section, we explain the experimental setup that is used throughout the paper in detail.

Evaluation Methods

In our methods, we generally apply two types evaluation methods: *agvx* and *ktx*. One measures how the algorithm performs in the top ranked objects and is described for each dataset separately. The second one compares the algorithm ranking x to a ground truth g ranking given by an external measure of prominence using the *Kendall-tau* measure.

Kendall-tau measure is given by the number of pairs with the same ordering minus the number of pairs with flipped ordering, all divided by the total number of pairs compared. A Kendall-tau of 1 corresponds to identical orderings whereas -1 corresponds to full reversal. This computation is complicated by the fact that the ranking induced by some external measures of prominence (like h-index) contain a large number of ties. In contrast, most of the tested algorithms (with the exception of indegree) operate on real number ranges and produce very few ties, if any. As a result, whenever the algorithm reports a_1 is ranked higher than a_2, when in fact the the actors are tied in the ground truth, we would like to treat this as neither an agreement or a disagreement. An exception to this rule is the indegree and centrality algorithms which can produce a lot of ties where breaking a tie is to be considered a slight disagreement (penalty of 0.5).

In particular, suppose we are given ranked lists a for an algorithm ranking and g for a ground truth ranking. Both a, g may contain ties. In case of ties, objects are given the same rank. The next object is given the next rank that would be available if there were no ties. For example, objects with scores $0.1, 0.2, 0.2, 0.3$ are given ranks $1, 2, 2, 4$. Given these two ranked lists, the kendall-tau measure $kt(a, g, p1, p2)$ takes as input two penalty measures $p1, p2$ and is computed as follows:

Algorithm 2 kt $(a, g, p1, p2)$

```
1: Input: Algorithm $a$ & ground truth $g$ rankings of objects, penalties $p1, p2$
2: $Agree = Disagree = NumCompared = 0$
3: **for** all distinct pairs $(o_1, o_2)$ of objects that are both ranked by $a$ and $g$ **do**
4:     $NumCompared++$
5:     **if** $(o_1, o_2)$ tied in $a$, but not in $g$ **then**
6:         $Disagree += p1$, $Agree += (1 - p1)$
7:     **else if** $(o_1, o_2)$ tied in $g$, but not in $a$ **then**
8:         $Disagree += p2$, $Agree += (1 - p2)$
9:     **else if** $(o_1, o_2)$ ranked the same in $a$ and $g$ **then** $Agree += 1$
10:    **else** $Disagree += 1$
11:    **end if**
12: **end for**
13: return $(Agree - Disagree)/NumCompared$
```

Given this algorithm, whenever we are comparing iHypR to a ground truth value with ties, we use $p1 = 0.5$ and $p2 = 1$. Which means there is no penalty for iHypr to break a tie. However, as such pairs are counted in $numCompared$, there is an implicit penalty as such pairs do not contribute to agreement counts as well. For indegree and closeness, we use $p1 = p2 = 0.75$ and $p2 = 0.25$ which means that a tie from g broken by indegree is considered a penalty of 0.5 overall. This is a well known method of treating ties as half flips [Fagin et al. 2003].

Clustering Algorithms Used

SSDE. SSDE-Cluster [Magdon-Ismail and Purnell 2011] is the default clustering algorithm used by iHypR. This algorithm first embeds the input graph into metric space and then uses the Gaussian Mixture Model to find a prespecified number n of overlapping clusters.

SSDE-cluster also has a module for automatically determining the number of clusters based on how the clueter error drops relative to a null hypothesis random graph model. Based on this method, we find that (for example) 10 clusters are significant in DBLP, and within each cluster, 10 subclusters are significant, suggesting a natural 10-10 hierarchical clustering of the DBLP objects. This clustering roughly corresponds to 10 research areas and 10 sub-areas. For comparison we also used a single (no-hierearhical) clustering into 100 clusters. We thus use two possible methods to find clusters in SSDE:

— *Cluster the whole dataset.* We find total number n of clusters in SSDE using the methodology above and cluster the data into n overlapping clusters. This gives us a top view of the data.
— *Cluster in two levels.* We first find the total number n of clusters in SSDE. Then, we first cluster the dataset into $\sqrt{n}$ top level clusters. These clusters correspond to higher level communities in the data. For example, in the case of DBLP, this would correspond to research areas. Then, we rerun the SSDE for each cluster and find $\sqrt{n}$

clusters within each cluster. This gives us a total of n clusters, but taking into account that the optimization of the clustering criteria for the smaller clusters is local to each top cluster. This allows us to take into account differences in different communities. For example, in DBLP, certain communities are tightly connected and others are very loosely connected. For example, systems papers tend to cover a very large number of authors with fewer connections than papers in neural networking.

FC: Fast Community. FastCommunity [Clauset et al. 2004] finds densely connected subgroups of the graph and produces disjoint clusters of varying sizes and number depending on the dataset. It does not take as input the number of clusters expected in the output. Note that we use the same input graph as in Section 2.1 for FastCommunity.

Fastcommunity does not have any input parameters, our run produced 2638 and 1170 for DBLP object graphs using w_{aa} and w_{jac} respectively.

RC: RankClus. RankClus [Sun et al. 2009a] takes a dataset with predefined venues and partitions these venues into n disjoint clusters where n is an input to the algorithm. The intuition behind this algorithm is that it finds groupings of the venues that correspond to research areas. These groups are then used to rank the objects separately within each cluster. Note that we do not consider the ranking produced by this algorithm, but use the clusters produced by it. RankClus takes as input the DBLP graph, i.e. authors, papers and venues.

Note that there is no methodology for choosing the number of clusters for RC. Given the code provided to us was not able to deal with the full DBLP data, we have chosen to use instead the dataset that was originally used in the RankClus paper which excludes the authors with less than 5 papers and finds 20 clusters (RC20). We also found people with 10 or more papers and ran RankClus for 50 clusters (RC50).

Note that we also tried the CFinder algorithm [Clauset et al. 2004], but were unable to run it due to the memory requirements of this algorithm. Below, we summarize the properties of the algorithms used in this paper.

Algorithm	Does not need num clusters?	Does not need hyperedges?	Overlapping clusters?
SSDE	✓	✓	✓
FC	✓	✓	×
RC	×	×	×

Datasets Tested

DBLP dataset. DBLP was originally developed for the Database and Logic Programming community, but is now extended to a large subset of Computer Science. The venues in DBLP contain conferences, journals, and book chapters. In our tests, we used the RDF tagged version DBLP from April 2008 called SwetoDBLP [6]. Our dataset contains 1,004,959 publications, 615,416 authors, 4,718 books chapters from 1,335 books, 386,481 journal articles and 613,760 conference papers. From this dataset, we extract the largest connected component and use it in all our tests. Note that unless otherwise noted, all our tests refer to this dataset. Furthermore, a venue is considered to be a specific year of a conference or a journal. Each book is considered a venue as well.

A subset of the conference papers and books (495,159 total) have an additional tag with the DBLP internal URL of the proceedings the paper was included in (we will call this set of papers *III: isIncludedIn*). The remaining conferences do not seem to have a specific proceedings URL. This appears to be more of a data curation problem, for example AMAI appears in both sets, with or without proceedings. However, the

[6]http://archive.knoesis.org/library/ontologies/swetodblp/

dataset III excludes all journals which is a significant difference between ALL and III. We use this dataset in one of our experiments given in Section 4.1 and refer to it as AOT:conf.

For external prominence, we use consider two different measures:

Hindex values. We use citations, especially, h-index as an external method of prominence for authors [Hirsch 2005]. Note that DBLP contains some citation data, but this data tends to be sparse. We do not use this data in our prominence computations or in h-index computation. Instead, we collect h-index values directly from Microsoft Academic Search (MAS) [7]. We remove all authors with h-index value less than three to reduce noise in our data. The final MAS h-index dataset contains $84,804$ distinct authors.

TC10 values. H-index and its variations are used frequently as a measure of prominence. As h-index is insensitive to the actual number of citations, it does not help distinguish between researchers who are extremely prolific and have done seminal work. For example, an author who has two seminal papers may have the same h-index as another with two papers of at least two citations. While the g-index tries to capture this value, it is not sufficiently fine tuned for our purposes. A person with 10 papers need to have at least 100 citations for their top paper. However, it is possible that the top cited paper has 1,000 citations. Both cases will appear the same under the g-index. On the other hand, for a person with 50 papers, the top paper needs to have 2,500 citations which would be an extremely high value to capture. To be able to find people who have produced very highly cited papers, we introduce a new measure: **tc-10**, which computes the average number of citations an author got for her top 10 most cited papers.

As MAS does not contain detailed citation information, we collect citation data using Google Scholar. We query Google Scholar for randomly picked author names from DBLP and limit the search fields to Engineering, Computer Science, and Mathematics. We then parse the returned html files to compute h-index and tc-10values. Since the only information we submitted to Google Scholar is author names, the returned information may contain a lot of noise especially when the name is too abbreviated or common (e.g. “J. John”).

To reduce noise in our computed tc-10 values, we compute the absolute difference between the h-index values computed using Google scholar and those from the MAS dataset. We then selected the subset of authors with difference of at most 10 and use the tc-10 values of these authors. The final tc-10 dataset has $56,058$ authors.

Enron Dataset. We have used the Enron email data set [8] that contains mails between employees of Enron. In our tests, we have not considered the content of the emails, only sender and receiver information. We have removed emails that correspond to broadcast or almost broadcast messages with more than 20 recipients. In our dataset, we have a total of 21,644 messages between 156 Enron employees. The messages (objects) link people (actors): the sender to the receiver. In contrast with the DBLP dataset where the collaboration is not directed, the direction of the messages is important. It is questionable whether emails have intrinsic value that can be measured. We assume that most emails do not have value themselves but they show the relationship between the employees. Unlike DBLP, there are no true hyperedges, so we can only use the clusters for this purpose. As mentioned earlier, we use the positions of the 115 employees in the organizational hierarchy used in previous literature as an external measure of

[7]http://academic.research.microsoft.com/
[8]www.cs.cmu.edu/∼enron/

prominence [Diesner et al. 2005]. For Enron, we use 9 clusters, 3 at the top level and 3 at the lower level.

IMDB. Finally, we use the IMDB [9] database containing information about movies (objects) and actors. Multiple people starring in a movie corresponds to a collaboration between the actors. However, casting decisions are not voluntary. In DBLP, an author can decide to write with another author. In Enron, a person can decide to send an email to another. However, actors do not necessarily decide with whom they will act together. This feature sets IMDB apart from the previous two datasets.

Each actor has prominence and each movie has prominence. However, there are many different metrics for measuring the prominence of movies and actors. Using the quality of the movie is one possible idea, but quality tends to be subjective as evidenced by the number of different reviews found for a movie. Furthermore, we do not have historical data for review aggregation sites like rottentomatoes.com. But, we use the ratings in IMDB for this. We also considered using the popularity of the movies as an indicator of the prominence of the actors in them. One way to achieve this is to use the movie gross information. However, literature in cultural markets [Salganik 2007] shows that it is very hard to predict how successful a movie is going to be while it is being made. As a result, one cannot argue that the movie gross is a strong indicator of the actors' prominence. Movies of many prominent actors fail after all. To overcome this problem, we use the movie budget as an indicator of the prominence of the actors in the movie. The budget indicates how much the movie industry believes the actors are worth and hence is a more reliable measure of the prominence of the actor in the movie industry. IMDB contains movie budget data for an increasing number of the movies in the later decades especially VI.

Table VI. Movies with budget, gross and rating information information in IMDB

Decade	#Movies	#Movies w/ budget	#Movies w/ gross	#Movies w/ rating
1930s	10285	411	72	5789
1940s	7359	321	81	5426
1950s	5926	355	114	4833
1960s	3787	348	122	3325
1970s	3599	431	284	3268
1980s	3834	708	1590	3681
1990s	6019	1487	2366	5731
2000s	18633	8089	3080	13059

In our tests, we use the full IMDB dataset. As this dataset contains adult movies, Hollywood movies and foreign movies from three different movie industries, this makes prominence computation particularly hard for many algorithms. In the earlier decades, the movie and budget information solely consists of Hollywood movies. However, in later years, all three sets of movies are increasingly intermixed. In this dataset, we focus on the stars and the director of the movies. This is based on the assumption that the top actors and the director generally appear in the movie advertisements and hence are the most relevant factors in judging the marketability of the movie. We therefore extract the stars of the movies (and the directors) by picking the top three actors in the order of billing. We also add the director(s) of the movie to the analysis.

For external evaluation using movie budget info, we consider movie budget for each decade separately. This helps solve the problem that the movie budget information is

[9] www.imdb.com

Algorithm	avgh	avgt	kth	ktt
iHypR$_{AOH}$	42.9	456.66	0.44	0.28
iHypR$_{AT}$	41.55	361.59	0.39	0.26
PG_{AOT}	35.9	332.69	0.44	0.26
PG_{AOH}	35.9	338.9	0.44	0.26
$PG_{A_{norm}}$	38.3	414.31	0.39	0.22
iHypR$_{AOT}$	40.55	307.94	0.40	0.26
PG_{AO}	32.5	352.5	0.43	0.26
$PG_{A_{aa}}$	33.85	306.53	0.42	0.23
iHypR$_{AH}$	26.95	378.01	0.4	0.23
$Deg_{A_{norm}}$	35	408.33	0.27	0.24
$PG_{A_{log}}$	39.75	222.13	0.43	0.22
$InDeg_{A_{norm}}$	36.05	286.17	0.28	0.24
$Hits_{AO}$	11.8	55.31	0.32	0.19
$Hits_{AH}$	14.2	68.37	0.34	0.15
$Hits_{A_{norm}}$	6.2	33.67	0.24	0.15

Fig. 13. The real values for avgh, avgt, kth and ktt for Figure 3 based on the DBLP dataset.

a one time measure, it does not accumulate over time. So, considering it for the lifetime of a movie is not appropriate as we did for citations in DBLP. For each decade, we construct the movie to movie graph as in Section 2.1 using w_{aa} and compute hypergraphs. Our results indicate that IMDB does not exhibit subcommunity structure and using subcommunities slightly hurts performance. So, we use 16 top level clusters in our tests as a result.

Given the movie budget information, we assume the value of an actor is equal to the average budget of the movies they starred in. However, the movie budget values between different years is not comparable and is very sensitive to outliers, i.e. very high budget movies. To overcome this problem, we normalize the movie budget information for each year between 0 and 1 as follows. We first partition the movies into years, then rank them by its budget within the decade it belongs to. Finally, we assign a value to the movie (normalized movie value) by the equation:

$$mv(i) = \frac{k - r(i)}{k - 1}$$

Where $r(i)$ is the ranking of movie i, and k is the total number of movies in that year. Note that movies with highest gross in a year will get 1 and the last movie and movies with no gross information will get 0. In the case that only one movie in a year, we assign 1 to the movie. The value of an actor is then given by the average normalized value of the movies they starred in.

Performance of various ranking algorithms on the DBLP dataset

In Figure 13, we provide the true performance figures (not just the regret values) of various ranking algorithms on the DBLP dataset. The regret values based on these numbers were given earlier in Figure 3.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Jiawei Han for sharing the RankClus code with us.

REFERENCES

ADALI, S., LU, X., MAGDON-ISMAIL, M., AND PURNELL, J. 2011. Prominence ranking in graphs with community structure. In *International AAAI Conference on Weblogs and Social Media (ICWSM'11)*.

ADAMIC, L. AND ADAR, E. 2003. Friends and neighbors on the web. *Social Networks 25*, 211–230.

ADAMIC, L. A., LAUTERBACH, D., TENG, C.-Y., AND ACKERMAN, M. S. 2011. Rating friends without making enemies. In *Proceedings of the AAAI Conference on Social Media and Weblogs (ICWSM 2011)*.

BALMIN, A., HRISTIDIS, V., AND PAPAKONSTANTINOU, Y. 2004. Objectrank: Authority-based keyword search in databases. In *Proceedings of the Thirtieth international conference on Very large data bases*. Vol. 30. 564–575.

BAO, S., XUE, G., WU, X., YU, Y., FEI, B., AND SU, Z. 2007. Optimizing web search using social annotations. In *Proceedings of the 16th international conference on World Wide Web*. WWW '07. ACM, 501–510.

BORODIN, A., ROBERTS, G., ROSENTHAL, J., AND TSAPARAS, P. 2005. Link analysis ranking: Algorithms, theory, and experiments. *ACM Transactions on Internet Technology 5,* 1, 231–297.

BRIN, S. AND PAGE, L. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the ACM WWW Conference*. 107–117.

CLAUSET, A., NEWMAN, M., AND MOORE, C. 2004. Finding community structure in very large networks. *Phys. Rev. E 70,* 6, 066111.

DIESNER, J., FRANTZ, T., AND CARLEY, K. 2005. Communication networks from the enron email corpus, it's always about the people. enron is no different. *Computational and Mathematical Organization Theory 11*, 201–228.

FAGIN, R., KUMAR, R., AND SIVAKUMAR, D. 2003. Comparing top k lists. *SIAM J. Discrete Mathematics 17,* 1, 134–160.

FAUST, K. AND WASSERMAN, S. 1994. *Social Network Analysis: Methods and Applications*. Cambridge University Press.

GIBSON, D., KLEINBERG, J., AND RAGHAVAN, P. 2000. Clustering categorical data: an approach based on dynamical systems. *The VLDB Journal 8,* 3-4, 222–236.

GYONGYI, Z., GARCIA-MOLINA, H., AND PEDERSEN, J. 2004. Combating web spam with trustrank. In *Proceedings of the 30th International Conference on Very Large Data Bases*.

HAVELIWALA, T. 2002. Topic-sensitive pagerank. In *Proceedings of the ACM WWW Conference*. 517–526.

HIRSCH, J. 2005. An index to quantify an individual's scientific research output. *Proc. of the National Academy of Sciences 46*, 16569–16572.

HOTHO, A., JÄSCHKE, R., SCHMITZ, C., AND STUM, G. 2006. Information retrieval in folksonomies: Search and ranking. In *The Semantic Web: Research and Applications, volume 4011 of LNAI*. Springer, 411–426.

JEH, G. AND WIDOM, J. 2002. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '02. ACM, 538–543.

JONES, B., WUCHTY, S., AND UZZI, B. 2008. Multi-university research teams: shifting impact, geography, and stratification in science. *Science 322*, 1259–1262.

KLEINBERG, R. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM 46,* 5, 604–632.

LEMPEL, R. AND MORAN, S. 2001. Salsa: The stochastic approach for link-structure analysis. *ACM Transactions on Information Systems 19,* 2, 131–160.

MAGDON-ISMAIL, M. AND PURNELL, J. 2011. Ssde-cluster: Fast overlapping clustering of networks using sampled spectral distance embedding and gmms. In *IEEE International Conference on Social Computing*.

NEWMAN, M. E. J. 2003. Mixing patterns in networks. *Physical Review E 67*.

NIE, Z., ZHANG, Y., WEN, J.-R., AND MA, W.-Y. 2005. Object-level ranking: bringing order to web objects. In *Proceedings of the 14th international conference on World Wide Web*. WWW '05. ACM, 567–574.

RIVERA, M., SODERSTROM, S., AND UZZI, B. 2010. Dynamics of dyads in social networks: Assortative, relational, and proximity mechanisms. *Annual Review of Sociology*, 91–115.

SALGANIK, M. J. 2007. Success and failure in cultural markets. Ph.D. thesis, New York: Department of Sociology, Columbia University.

SUN, Y., HAN, J., ZHAO, P., YIN, Z., CHENG, H., AND WU, T. 2009a. Rankclus: integrating clustering with ranking for heterogeneous information network analysis. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. 565–576.

SUN, Y., YU, Y., AND HAN, J. 2009b. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '09. ACM, 797–806.

WEST, D. B. 2000. *Introduction to Graph Theory (2nd Edition)*. Prentice Hall.

WUCHTY, S., JONES, B., AND UZZI, B. 2007. Commentary: Why do team-authored papers get cited more? *Science 317*, 1496.

XI, W., ZHANG, B., CHEN, Z., LU, Y., YAN, S., MA, W.-Y., AND FOX, E. A. 2004. Link fusion: a unified link analysis framework for multi-type interrelated data objects. In *Proceedings of the 13th international conference on World Wide Web*. WWW '04. ACM, New York, NY, USA, 319–327.

XIAOMING LIU, JOHAN BOLLEN, M. L. N. AND DE SOMPEL, H. V. 2005. Co-authorship networks in the digital library research community. *Information Processing & Management 41*, 1462–1480.