Can we measure the impact of a database?

Peter Buneman
University of Edinburgh
Edinburgh, UK
opb@ed.ac.uk

Dennis Dosso
Seco Mind
Padua, Italy
dennis.dosso@secomind.com

Matteo Lissandrini
University of Verona
Verona, Italy
matteo.lissandrini@univr.it

Gianmaria Silvello
University of Padua
Padua, Italy
gianmaria.silvello@unipd.it

He Sun
University of Edinburgh
Edinburgh, UK
h.sun@ed.ac.uk

ABSTRACT

In disseminating scientific and statistical data, on-line databases have almost completely replaced traditional paper-based media such as journals and reference works. Given this, can we measure the impact of a database in the same way that we measure an author's or journal's impact? To do this, we need somehow to represent a database as a set of publications, and databases typically allow a large number of possible decompositions into parts, any of which could be treated as a publication.

We show that the definition of the *h*-index naturally extends to hierarchies, so that if a database admits some kind of hierarchical interpretation we can use this as one measure of the importance of a database; moreover, this can be computed as efficiently as one can compute the normal *h*-index. This also gives us a decomposition of the database that might be used for other purposes such as giving credit to the curators or contributors to the database. We illustrate the process by analyzing three widely used databases.

CCS CONCEPTS

• **Information systems** → **Database utilities and tools**; **Novelty in information retrieval**; **Ontologies**; **Clustering and classification**.

KEYWORDS

Data Citation, *h*-index, Scientific and Curated Databases

ACM Reference Format:
Peter Buneman, Dennis Dosso, Matteo Lissandrini, Gianmaria Silvello, and He Sun. 2024. Can we measure the impact of a database?. *To appear in the Communications of the ACM.*

1 INTRODUCTION

It is almost self-evident that databases exist to publish data, and this is undoubtedly the case for scientific and statistical databases, which have largely replaced traditional reference works. Database and web technology has led to an explosion in the number of databases that support scientific research for obvious reasons: databases provide faster communication of knowledge, they hold larger volumes of data, they are more easily searched, and they are both human and machine-readable. Moreover, they can be developed rapidly and collaboratively by a mixture of researchers and curators. For example, more than 1500 curated databases are relevant to molecular biology alone [12]. The value of these databases lies not only in the data they present but also in how they organize that data.

If we want to measure the impact of a database, can we use its organization to treat it in the same way that we treat any other publishing agent, such as a journal or an author?

In the case of an author or journal, most bibliometric measures are obtained from the citations to the associated set of publications. For a database, there are typically many ways of decomposing it into publications, so we might use its organization to guide in the choice of decompositions. We shall show that, when the database has a hierarchical structure, there is a natural extension of the h-index that works on hierarchies.

Although the main results presented in this paper are the evaluations of the *h*-indexes of some well-known databases, this was not the original motivation. One of the authors was involved in a project [6] to automatically generate a set of conventional papers to give credit – as authors – to the 1000 or so researchers who had contributed to a database [10]. By creating one publication for the whole database – as might happen with *data papers* which periodically publish data summaries and are citation proxies for databases [8], we generate a document with an unhelpfully huge number of authors. Also, these authors receive only one additional publication credit, regardless of whether they contributed to numerous sections of the database or just a single part. On the other hand, generating thousands of documents, one for each "object" in the database, is almost equally useless. While the authors are now associated with their areas of expertise, they are unwittingly guilty of having minimal publishing units. The tension between those two extremes underlies the rationale for the *h*-index [11], so the obvious question is can we extend the *h*-index to work on databases and – given that there is a hierarchical structure – is there a natural extension of the *h*-index to hierarchies? We then have at least one non-trivial measure of the importance of a database. However, even if this measure is not of interest, the decomposition that produced it may well be of interest, perhaps as a starting point to the curators, who want to find a useful set of publications to associate with the

database for the benefit of the contributors or curators who would like to have their work properly cited.

Hierarchies are used frequently in curated databases. As in the three examples we use in this paper [10, 16, 19], they are based on some kind of hierarchical classification scheme, taxonomy or ontology. Moreover, data sets based on a file system or data format such as JSON or XML have an intrinsic hierarchical structure (we shall refer to all of these as databases). We also note that, when two very similar papers are published (e.g., a preprint and its final peer-reviewed version), citation analyzers such as Google Scholar have an option to transfer citations from one paper to the other (the parent). Given that a paper can have at most one parent and that the parent relationship is acyclic, there is at least a partial hierarchy (a set of tree like structures) already present in the data structure maintained by the software.

Given a hierarchy, how do we use it to limit the possible decompositions into sets of publications? Consider hierarchy in the DrugBank database shown in Figure 1. In this database, citations are only to the terminal nodes or leaves of the tree. If we want a citation count for some higher-level node, we would use *the sum of the citation counts of the leaves below that node*, much as we would take the citation count for an issue of a journal to be the sum of the citation counts of the papers in that issue. Now, we propose to use a subset of these nodes as a possible decomposition; however we cannot use an arbitrary subset, because it allows double-counting of citations if we allow a node and one of its ancestors to appear in the same decomposition. Thus we restrict our attention to *antichains* of nodes: an antichain is a set of nodes in which no node is an ancestor of another node in that set. Looking at the Linnean-style stratification in Figure 1, the kingdom, superclass and class nodes each constitute antichains as do the drug nodes (the subclass nodes can have other subclass nodes as ancestors and do not). Also, the root node represents the database as a whole, and the leaf nodes represent the individual drugs, which both constitute hierarchies.

Now, the h-index, which is used almost universally to measure an author's output and often the importance of a journal; it is one of the few metrics that measure both the productivity and the citation impact of authors [17]. It is defined as follows:

> Given a set S of publications, its h-index is the largest number n for which there is a subset of S of size n in which each publication has at least n citations.

The extension to hierarchies is immediate:

> Given a *hierarchy* H of publications, its h-index is the largest number n for which there is an *antichain* in H of size n in which each publication has at least n citations.

We will formalize this in the following section, but we should make some important observations immediately. First, when the hierarchy is "flat" (there is no ordering), the two definitions coincide. Second, although a set has an exponential number of subsets, its h-index can be efficiently computed by sorting the set. Similarly, a hierarchy can have an exponentially large number of antichains, and we will demonstrate that, by using the appropriate data structures, the h-index of a hierarchy can be evaluated with the same efficiency. This is crucial if we are going to apply it to large databases.

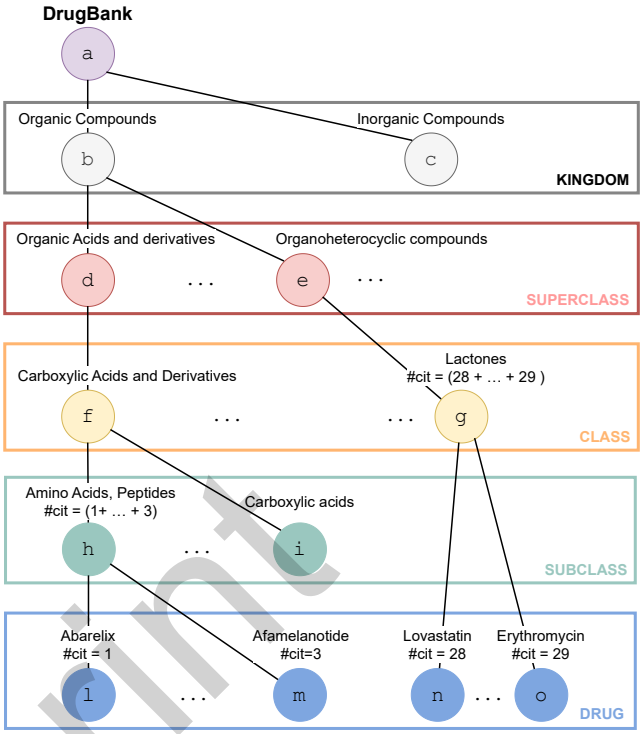

Figure 1: Partial representation of the Drugbank hierarchical structure. The web pages associated with the nodes are shown on the left. In Drugbank only the leaves of the hierarchy are directly cited; e.g., we can see that the citations of "Lactones" is the aggregation of the citations to the drugs (we report only Lovastatin and Erythromycin) belonging to the class.

Third, using a hierarchy to constrain the possible decompositions is essential. For example, based on a reduction from the Partition problem, one can show that the more general problem of finding the maximal h-index under an arbitrary partition is strongly NP-hard [9], and there are algorithms and complexity research on improving and maximizing the citation indices (e.g., h-index, g-index, and $i10$-index) under different merging rules [9, 14, 18].

Finally, much of this research can be seen as an attempt to "game" the h-index — finding some merging strategy that enhances one's h-index. If, for a hierarchical organization, we equivalently define the h-index of a hierarchy as the maximal h-index of any antichain within the hierarchy, then it might seem as though we are also engaged in gaming. However, this is not the case; we are merely employing what appears to be the most natural extension of the h-index definition to hierarchies.

In the following sections, we first formalize the problem and develop some simple results on antichains that allow that the maximal h-index to be computed efficiently. Following this, we give three examples of databases from which we can extract an h-index and look at some of the details of their hierarchical organization. We conclude with some speculation on how these results might be further developed.

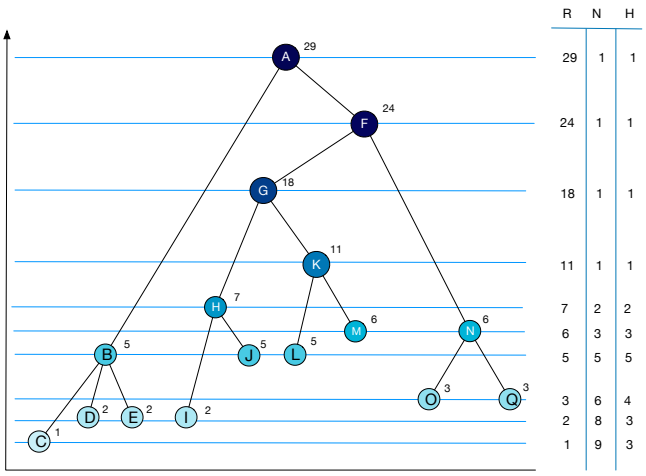

Figure 2: A hierarchical citation structure in which the level of each node is determined by its rank – the number of citations to it. Columns: R – the level; N – the length of the maximal l-antichain at that level; and H – the h-index for that antichain.

There are two major caveats to what we have to offer. The first is that we are not going to comment on the justification or fairness of the h-index and its variants [4, 21]; we only observe that it is almost universally adopted in measuring the impact of both authors and journals and – as we have noted above – it may be useful in helping to decompose a database into citable units.

Second, the results we present here are preliminary. As we have already remarked, the practice of data citation is still in its infancy; people still fail to cite databases or cite them improperly. For comparison, we have included the results of using URLs as citations. *Hence, the results we give should in no way be taken as an accurate measure of the importance of the databases we examine. They demonstrate appropriate techniques and the feasibility of measuring the* h-index *at scale.*

2 THE *H*-INDEX OF A HIERARCHY

2.1 Hierarchies and antichains

Our starting point is that we can model a database as a hierarchy together with a citation count for each node in the hierarchy. Our goal is to select a subset of nodes, aggregate the citations associated with those, and thus compute the h-index of such set of nodes. In choosing a subset of nodes, we cannot double-count citations. For example, in Figure 1 if we decide – to compute an h-index – that the *Lactones* node (node g at the "classes" level) is to be a publication, the citations to that node include all the citations to drugs beneath that node, so we cannot use a citation to one of those drugs as a citation to some other node. This means that we limit our candidate sets to those in which no node is an ancestor of another, or *antichains*.

To illustrate this, Figure 2 shows a hierarchy in which the root $\{A\}$ constitutes an antichain, as does $\{B, G, N\}$ and also the leaves. In this diagram, the level of a node is determined by its citation count or *rank*. Even though the number of antichains in a hierarchy can be exponentially large, to compute the h-index, we can use the rank to cut down the number of antichains we need to examine. For this, we need a small amount of formal development.

Definition 2.1. A *hierarchy* is a partial order $(H, \preccurlyeq)$ such that for all $a, b, c \in H$, $a \preccurlyeq b$ and $a \preccurlyeq c$ implies that $b \preccurlyeq c$ or $c \preccurlyeq b$.

This definition of hierarchy ensures that a node can have at most one parent. We note that n' is the *parent* of a node n if it is the smallest node $n' \in H$, distinct from n, such that $n \preccurlyeq n'$.

Definition 2.2. A set $A \subseteq H$ is called *antichain* if, for any distinct pair $n_1, n_2 \in A$, neither $n_1 \preccurlyeq n_2$ nor $n_2 \preccurlyeq n_1$.

Definition 2.3. A *ranked hierarchy* $(H, \preccurlyeq, r)$ is a hierarchy $(H, \preccurlyeq)$ together with a rank (or level) function $r : H \to \mathbb{N}$, such that for all $n_1, n_2 \in H$, if $n_1 \preccurlyeq n_2$ then $r(n_1) \leq r(n_2)$.

Given a subset S of H, we will use the term *minimal elements* of S to refer to those nodes $n \in S$ for which there is no $n' \in S$, distinct from n, such that $n' \preccurlyeq n$. That is, the minimal elements of S are the lowest nodes in the hierarchy belonging to S.

Definition 2.4. Given a ranked hierarchy $(H, \preccurlyeq, r)$, an l-*antichain* in H is an antichain A such that for all $n \in A$:

(1) $r(n) \geq l$,
(2) $\forall\, n' \in H$ if $n' \prec n$, then $r(n') < l$.

In other words, an l-antichain is a set in which each node has rank at least l and all its children have rank less than l. In Figure 2 the set $\{H, K\}$ is a 7-antichain and the leaves form a 1-antichain. Note that a leaf (a node with no children) with rank at least l could be a member of a l-antichain.

We shall use the term *rank-minimal* to describe the elements of a set that are minimal with respect to the rank function r, i.e., those elements $n \in S$ for which $r(n)$ is minimum for S.

The following result guarantees that, given an antichain with minimal rank l, it is possible to find an l-antichain which has no fewer elements.

Proposition 2.5. *Let A be an antichain in a ranked hierarchy $(H, \preccurlyeq, r)$ with a rank-minimal node of rank l. Then, there exists an l-antichain $A' \subseteq H$ such that $|A'| \geq |A|$.*

The proof is straightforward and given in the additional material (A.1); one constructs the antichain by a process that is similar to the top-down algorithm, which we describe shortly.

From this, we can obtain the main result. The h-index of an antichain $A \subseteq H$ in a ranked hierarchy $(H, \preccurlyeq, r)$ is defined as $h(A) = \max\{|S| \mid\ S \subseteq A \wedge \forall n \in S. r(n) \geq |S|\}$.

Proposition 2.6. *For any antichain A in a ranked hierarchy, there is an l-antichain A' such that $h(A') \geq h(A)$.*

Proof. Let $l = h(A)$ be the h-index of A. Then there is a subset $S \subset A$ for which $r(A) \geq |\, S \,| = l$ for all $n \in S$. A rank-minimal element of S will have rank $l' \geq l$ and Prop. 2.6 gives us an l'-antichain S' with at least as many elements as S, i.e., S' has an h-index no less than l. □

The importance of this result is that we only have to search maximal l-antichains to find the h-index of any antichain in H. The

number of such antichains is not greater than the number of nodes, which guarantees us at least a polynomial time algorithm. In fact, by pursuing a top-down approach, we can obtain a very efficient algorithm which, once the hierarchy is constructed, will, in practice, only visit a subset of the nodes in the hierarchy.

2.2 A top-down algorithm

Consider the l-antichains in Figure 2. At the root, we have a 29-antichain A of length 1 and below it a 24-antichain F of length 1. Proceeding downwards, the first longer antichain is the 7-antichain $\{H, K\}$ of length 2. Further down we find the 5-antichain $\{B, J, L, M, N\}$ of length 5. This is the l-antichain of greatest h-index: it has 5 nodes, all with rank greater or equal to 5. As we proceed downwards, the level is strictly decreasing while the length of the antichains is non-decreasing. When the latter overtakes the former, we have just "overshot" the antichain that yields the maximal h-index, and we can stop. Note that we have stopped before examining all the nodes in the hierarchy.

In the following we use the notation $MIN_{\preccurlyeq}(S)$ for the minimal elements of a subset $S \in H$, i.e., an antichain. $MAX_{\preccurlyeq}$ is defined similarly. An l-antichain can thus be defined as $MIN_{\preccurlyeq}\{n \in H | r(n) \geq l\}$.

In order to compute successively lower antichains, the algorithm maintains, for decreasing values of l, the following two antichains:

(1) $LCHAIN = \min_{\preccurlyeq}\{n \in H | r(n) \geq l\}$, i.e., the l-antichain of maximum cardinality among all l-antichains;
(2) $DCHAIN = \max_{\preccurlyeq}\{n \in H | r(n) < l\}$, i.e., the antichain of nodes that are maximal (with respect to $\preccurlyeq$) in the set of nodes with rank less than l.

These two antichains have the following properties:

(1) the children of any element in *LCHAIN* are contained in *DCHAIN*;
(2) the next lowest rank $l' < l$ of any node in the hierarchy is possessed by at least one node in *DCHAIN*;

Also, we observe that: i) a node n is in *LCHAIN* iff $r(n) \geq l$ and for all children m of n, $r(m) < l$; ii) a node n is in *DCHAIN* iff $r(n) < l$ and the parent p of n is such that $r(p) \geq l$.

Algorithm 1 details the procedure used to find the antichain that guarantees the maximum h-index. On lines 3 and 13, the nodes with the highest rank (i.e., citation count) l are found in the *DCHAIN*. These nodes are removed from the DCHAIN (line 7) and added to the *LCHAIN* (line 8). The children of these nodes are then added to *DCHAIN* (lines 9 and 10). Then, in lines 11 and 12, all the non-minimal nodes in *LCHAIN* are removed, making it a proper l-antichain. The process continues until the cardinality of *LCHAIN* is bigger than the highest rank l in *DCHAIN*. At this point, each node in *DCHAIN* has a rank $\leq l$. Thus even if it were to be moved from *DCHAIN* to *LCHAIN* there would not be an increase in the value of the h-index for *LCHAIN*. At this point, the algorithm can stop.

2.3 Algorithm discussion

Referring to Algorithm 1, the most obvious thing to do is to use heaps to implement *DCHAIN* and *LCHAIN*. The heap values for each node *DCHAIN* should be the node rank, while the values for those in *LCHAIN* should be the maximal rank of the children of the node, which can be computed at line 9. With these representations, the "search" operations on lines 3, 6, 11, and 13 all incur unit cost, while the insert or delete operations on lines 7, 8, 10, and 12 are all $O(\log n)$ in the length of the antichain. This gives a bound of $O(n \log n)$ in the hierarchy size for the whole algorithm.

A further observation is that at line 9, we need only consider child nodes for inclusion if their rank is greater than the current length of *LCHAIN*; if this is not the case, they cannot participate in the final antichain, and they together with their descendants in the hierarchy may be safely ignored. In describing the results for the various databases in Section 3 we have included two figures: visited (vis.) – the number of nodes whose rank was interrogated and digested (dig.) – the number of nodes that entered *DCHAIN*. The algorithm's running time will be bounded by a constant in vis. and a constant times $n \log n$ in dig. In the measurements for a whole hierarchy (not the truncated ones where we cut off the hierarchy at a certain level), the digested nodes accounted for less than 10% of the total.

Another observation is that the algorithm requires that the set of nodes as input is a hierarchy, and thus that this hierarchy is built before this algorithm is run. Finally, we note that in the degenerate case of a hierarchy made of only leaves, the algorithm requires $O(n \log n)$ time to create the *LCHAIN* heap.

Algorithm 1: Finding l-minimal antichains

Input : roots: a list of the roots of the hierarchy;
children: a function that gives the list of children of a node;
Output : An antichain with maximal h-index

```
1  LCHAIN ← ∅
2  DCHAIN ← roots
3  l ← max{r(n)|n ∈ DCHAIN}
4  while |LCHAIN| ≤ l do
5      oldLCHAIN ← LCHAIN.copy()
6      while ∃n ∈ DCHAIN such that r(n) = l do
7          DCHAIN ← DCHAIN − {n}
8          LCHAIN ← LCHAIN ∪ {n}
9          foreach c ∈ children(n) do
10             DCHAIN ← DCHAIN ∪ {c}
11     while ∃n ∈ LCHAIN such that ∃c ∈ children(n) such that
           r(c) ≥ l do
12         LCHAIN ← LCHAIN − {n}
13     l ← max{r(n)|n ∈ DCHAIN}
14 return oldLCHAIN
```

3 EXPERIMENTAL ANALYSIS

Using this algorithm, we compute and analyze the computation of the h-index of three widely-used databases: Drugbank which we describe in Section 1; the IUPHAR/BPS Guide to Pharmacology [10] (GtoPdb), which is an open-access database on the biological targets of drugs and other molecules; and, the taxonomic

Table 1: H-index and statistics on Drugbank. "Full" (top row) – the calculations on the full hierarchy; "Leaves/drugs" (second row) – those on the flat hierarchy (only leaves).

hierarchy	***h*-index**	median	max	nodes	vis.	dig.
Full	69	100	260	11,803	3,743	417
Leaves/drugs	33	44	76	10,303	–	–
Subclass	63	105	1,369	837	541	276
Class	49	120	2,327	328	287	154

database curated by the National Center for Biotechnology Information (NCBI), which maintains one of the most comprehensive and best-organized collections of medical data [13]. The online additional material details all the information about these databases and how we collected the corresponding citation counts.

3.1 Drugbank

The results of the analysis on Drugbank are shown in Table 1, where *median* and *maximum* refer to the citation counts of the antichain that yields the given *h*-index (the median is sometimes given in reporting the *h*-index of a journal). The visited (vis.) and digested (dig.) columns refer to the discussion about Algorithm 1. Only a few nodes are leaves in the antichain for the entire hierarchy. The *Leaves/drugs* row shows the *h*-index for the "flat" hierarchy of the drugs themselves, that is, the *h*-index obtained considering only the drugs as a set of "publications" that receive citations, without considering the hierarchy of the database. The *Class* and *Subclass* rows show the results for the hierarchies obtained by removing all nodes below those strata. The table clearly shows the advantage to computing the *h*-index on a full hierarchy and second that constraining the antichains to a given stratum reduces the *h*-index. We also see that the algorithm calculates the *h*-index visiting approximately one-third of the nodes in the entire hierarchy.

3.2 GtoPdb

The IUPHAR/BPS Guide to Pharmacology (GtoPdb) is an open-access relational database with an accompanying website on the biological targets of drugs and other molecules. As shown in Figure 3, GtoPdb has a hierarchical structure. It combines the expertise of approximately 1,000 researchers worldwide and was an early stimulus for data citation [7] to give credit to the contributors. Here, however, we are using it to assess credit to the database as a whole.

A critical difference between Drugbank and GtoPdb is that in GtoPdb, there are citations to the intermediate nodes in the hierarchy. In fact the data for these "family" nodes often resembles short conventional papers with descriptive material and statistics on the targets (leaf nodes) included in the family. This raises the possibility that we might want to treat a family node as independent of the nodes it describes, and we introduce a *lifting* transformation that allows us to do this.

Table 2 presents the results for the GtoPdb database divided into the horizontal sections "Citations" and "Links". The first section corresponds to results obtained with citations extracted from the citing papers, while the second part is obtained by accounting for web links as citations.

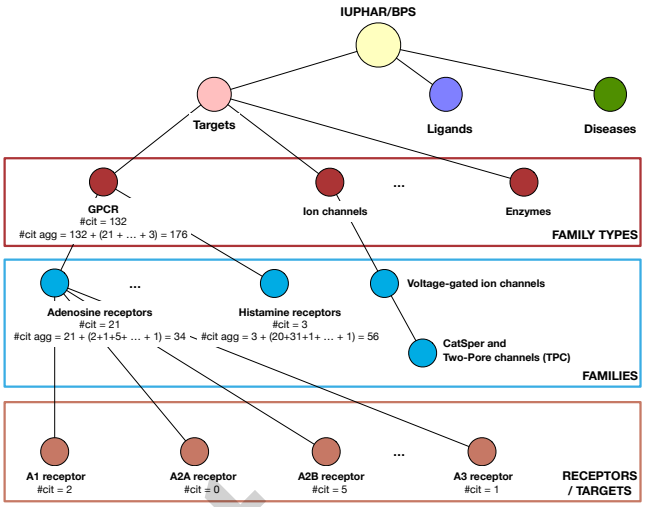

Figure 3: Part of the IUPHAR/BPS hierarchical structure. The root node represents the whole databases, and the family structure is not stratified as families may have subfamilies. All the nodes in the hierarchy can be independently cited; e.g., we show some sample citation numbers for the GPCR branch of the tree, where the internal nodes can receive direct citations (`#cit`) that could be also aggregated (`#cit agg`) with the citations of the child nodes.

As with Drugbank, the first two rows of each section in the table show the analysis on the full hierarchy and on the "flat" collection of nodes, which now contains a substantial number of cited intermediate (family) nodes together with the leaf target nodes. The "Families only" shows that little is lost by removing the leaves from the hierarchy, thus suggesting that many citations are also given to the intermediate nodes in the hierarchy. The meaning of the "Lifted hierarchy" row is discussed below.

It is also interesting to note that families of nearly all heights – where by height, we mean the maximum number of edges to a leaf node – figure in the antichain that provides the *h*-index for the full hierarchy. The figures are: 5 at height 3, 9 at height 2, 4 to leaves, and the remaining 37 to bottom level families at height 1.

Lifting. For reasons mentioned in the additional material, the Links section of Table 2, is based on highly skewed citation counts, and the results are almost meaningless. We have included it in this section because it shows an interesting phenomenon. We see – perhaps surprisingly – that computing the *h*-index on the full hierarchy gives a *lower* figure than that for the set of nodes with no structure (167 vs 172). It appears that creating a hierarchy on a set of publications, in this case the GtoPdb web pages, and summing the citation counts as we have proposed, does not necessarily increase the *h*-index with respect to the simple set of all the publications.

Consider now the hierarchy in Figure 4.a. Each node is annotated on the left with its number of "direct" citations and on the right with the sum of the number of citations to that node and to all of its descendants. It is easy to see that, for this first hierarchy, the longest antichain is the set $\{N_3, N_4\}$, which has an *h*-index of

Table 2: *h*-index calculations for GtoPdb. "Full" (top rows) are the calculations on the full hierarchy; "No" (second rows) are those on the flat hierarchy (only leaves); and "Lifted" (bottom rows) are those on the lifted hierarchy.

Cit. type	Hierarchy	***h*-index**	median	max	nodes	vis.	dig.
Citations	Full	55	77	276	4,079	1,185	275
	No	32	44	94	4,079	–	–
	Families	54	86	276	807	572	253
	Lifted	55	77	276	4,872	1,185	275
Links	Full	167	251	880	4,079	1,584	419
	No	172	253	909	4,079	–	–
	Families	135	298	3,013	807	588	211
	Lifted	179	265	909	4,872	1,604	468

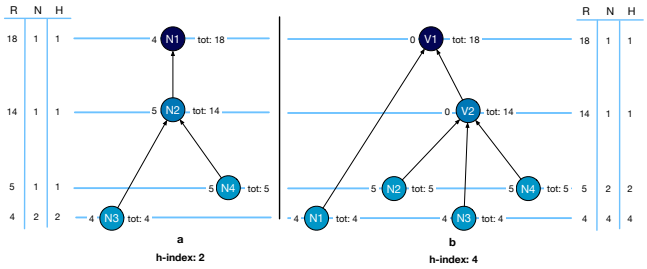

Figure 4: Example of one hierarchy (a) and its corresponding "lifted" version (b). Once again, R is the value of the rank at each level, N is the cardinality of the maximum antichain at each level, H is the *h*-index of that antichain. Below each hierarchy, its *h*-index is highlighted.

2. However, had we considered the set of nodes $\{N_1, N_2, N_3, N_4\}$ without a hierarchy and with a number of citations as the one obtained by direct citations to them (4, 5, 4, 5 respectively), its *h*-index would be 4. Therefore, in this example, the use of the hierarchy does not bring the highest *h*-index that could be obtained with the available publications.

On the other hand, had Figure 4.a been part of a larger hierarchy, it might have been advantageous to have a node such as N_1 with a relatively high rank. The implication is that, in a bigger hierarchy, new and longer antichains could have been found, where N_1 could contribute to the generation of a greater *h*-index. The problem is that in the computation of the *h*-index, as described in Section 2, the inclusion of a node in an antichain precludes the inclusion of any ancestor or descendant node in the same antichain. Thus, a descendant and ancestor node cannot be considered *independently* in the computation of the *h*-index.

In a case such as GtoPdb, however, the information associated with a family may complement the information associated with an object (child) in that same family. Therefore, It is reasonable to ask that both parent and child could be counted in the computation of the *h*-index.

To deal with this problem, we introduce a transformation called *lifting*: for *each* internal node N_i in the hierarchy, a surrogate node V_i is created. V_i becomes the parent of N_i along with all the children of N_i, and N_i becomes a leaf. N_i's rank is now determined only by its direct citations, and the new node V_i enters with a number of direct citations equal to 0 and a rank equal to the sum of its children's citations.

Figure 4.b shows the result of applying this transformation to Figure 4.a. Note how the set of all non-surrogate nodes $\{N_1, N_2, N_3, N_4\}$ now can be an antichain at rank 4.

After applying this transformation in the GtoPdb hierarchy, the results shown in Table 2("Lifted Hierarchy" row) show an *h*-index of 55 with data citations, and 179 with links.

We mentioned in the introduction that, for conventional papers, when two papers are similar, one may designate one of the papers as the parent, which will then receive all the citations for the child. Whether it is advantageous to combine the two depends on the authors. The advisor with a high existing h-index might want to do this, but the student with few publications might not. A lifted representation of the publication hierarchy and computing the *h*-index of an author on that hierarchy would keep both advisor and student happy!

3.3 NCBI Taxonomy

Among the databases maintained by NCBI is the taxonomy database [16], which contains "a list of names that are determined to be nomenclatural correct or valid (as defined according to the different codes of nomenclature), classified in an approximately phylogenetic hierarchy"[1]. The hierarchy contains some 2.3 million nodes or *taxons*, and for each node, the web interface displays incoming links from other NCBI databases and Pubmed, which can be considered as citations from conventional literature. The structure of NCBI is similar to that of IUPHAR/BPS as all nodes in the hierarchy are citable. There are approximately 3.2 billion incoming links for all nodes, of which 7.8 million are from Pubmed. For reasons described in the additional material, we confined our attention to the vertebrate and the animal sub-hierarchies. Vertebrates are a subset of animals, and animals account for approximately half the nodes in the complete hierarchy.

The results on the NCBI taxonomy hierarchy are shown in Table 3, where they are divided between the results from only the incoming links from Pubmed and the results obtained using all the available links. Unsurprisingly, the *h*-index for animals is substantially greater than that of its sub-hierarchy of vertebrates. Moreover, measuring an *h*-index using incoming links gives a substantially

[1] https://www.ncbi.nlm.nih.gov/books/NBK53758/

Table 3: *h*-index calculations for NCBI Taxonomy. "Given" (first rows) refers to the NCBI taxonomy without modifications; and "Lifted" (bottom rows) are those on the lifted hierarchy. "Pubmed" (top rows) are the calculations counting only the citations from Pubmed; and, "All" are the calculations counting Pubmed and incoming web links as citations.

Hierarchy	Refs from	Subtree	***h*-index**	Nodes	vis	dig
Given	Pubmed	animals	1,181	1,147,717	47,880	5,724
	Pubmed	vertebrates	681	112,271	17,503	3,133
	All	animals	3,794	1,147,717	489,692	16,712
	All	vertebrates	2,065	112,271	43,282	9,254
Lifted	Pubmed	animals	1,301	1,272,867	47,665	6,560
	Pubmed	vertebrates	758	131,464	17,700	3,638
	All	animals	3,887	1,272,867	494,769	18,284
	All	vertebrates	2,148	131,464	45,996	10,371

higher result and indicates the magnitudes we might see if we measure *h*-indexes using links rather than conventional citations. Lifted hierarchies noticeably raise the results.

4 CONCLUSIONS

This paper demonstrates that databases can be treated similarly to authors and journals for the purpose of measuring scholarly impact. We have shown a natural and efficient way to compute the *h*-index of a hierarchy of citable units and to apply this to computing the *h*-index of databases. Preliminary results show that, for some well-known curated databases, the *h*-index, when measured by citations from conventional literature, is comparable with that of journals. When measured by links from other databases, it can be substantially higher. The results are preliminary because data citation is not as widely practiced as it should be.

Throughout this paper, we have deliberately avoided any discussion of the rights and wrongs of using the *h*-index as a measure of the impact or quality of research output, which is a matter of continuing discussion [5, 21]. One empirical and theoretical observation is that the *h*-index is correlated with the square root of the total number of citations. In fact the ratio is in the neighborhood of 0.5. This holds for our results when considering citations *from the literature*. However, the results are very different when we measure – as for the NCBI taxonomy – an *h*-index for incoming links, whose total is in the billions. Here the ratio is an order of magnitude lower, indicating a skewed distribution that requires further investigation.

There are many more open problems. The most obvious is what one can do without an apparent hierarchy. If no classification scheme is available, some form of cluster analysis can nearly always find one. The more likely problem is that there are several classification schemes [3], and one may want to choose or combine them. Also, a classification scheme may not be hierarchical because a node may have more than one parent. Apart from coming up against the complexity issues described in Section 1, one has the prior problem of how the citations for a given node should be distributed among its parents to achieve a sensible ranking. It may be that the notion of disjunctive citations [20] could help with this. Finally, unlike conventional publications, databases evolve over time, yet for the purpose of citation, we want to treat them as conventional, immutable publications. This is a general problem with data citation and will also affect how we measure the impact of databases.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for constructive criticisms and Simon Harding, Jonathan Bard and Aris Filos-Ratsikas for their help. Huawei UK partially supported Peter Buneman. Dennis Dosso's work was carried out while he was with the University of Padua. Most of the research was conducted while Matteo Lissandrini was at Aalborg University. Gianmaria Silvello was partially supported by the HEREDITARY Project as part of the European Union's Horizon Europe research and innovation programme under grant agreement No GA 101137074. He Sun was supported by an EPSRC Early Career Fellowship (EP/T00729X/1).

REFERENCES

[1] Stephen PH Alexander, Arthur Christopoulos, Anthony P Davenport, Eamonn Kelly, Alistair Mathie, John A Peters, Emma L Veale, Jane F Armstrong, Elena Faccenda, Simon D Harding, et al. 2019. The Concise Guide to PHARMACOLOGY 2019/20: G protein-coupled receptors. *British journal of pharmacology* 176 (2019), S21–S141.

[2] Stephen PH Alexander, Eamonn Kelly, Alistair Mathie, John A Peters, Emma L Veale, Jane F Armstrong, Elena Faccenda, Simon D Harding, Adam J Pawson, Joanna L Sharman, et al. 2019. The Concise Guide to PHARMACOLOGY 2019/20: Introduction and other protein targets. *British journal of pharmacology* 176 (2019), S1–S20.

[3] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. 2000. Gene ontology: tool for the unification of biology. *Nature genetics* 25, 1 (2000), 25–29.

[4] L. Bornmann and H.-D. Daniel. 2007. What do we know about the h index? *Journal of the American Society for Information Science and Technology* 58, 9 (2007), 1381–1385. https://doi.org/10.1002/asi.20609

[5] Ricardo Brito and Alonso Rodríguez Navarro. 2021. The inconsistency of h-index: A mathematical analysis. *Journal of Informetrics* 15, 1 (2021), 101106.

[6] Peter Buneman, Greig Christie, Jamie A. Davies, Roza Dimitrellou, Simon D. Harding, Adam J. Pawson, Joanna L. Sharman, and Yinjun Wu. 2020. Why data citation isn't working, and what to do about it. *Database J. Biol. Databases Curation* 2020 (2020). https://doi.org/10.1093/databa/baaa022

[7] P. Buneman and G. Silvello. 2010. A Rule-Based Citation System for Structured and Evolving Datasets. *IEEE Data Eng. Bull.* 33, 3 (2010), 33–41.

[8] L. Candela, D. Castelli, P. Manghi, and A. Tani. 2015. Data Journals: A Survey. *Journal of the Association for Information Science and Technology* 66, 9 (2015), 1747–1762. http://dx.doi.org/10.1002/asi.23358

[9] Bart de Keijzer and Krzysztof R. Apt. 2013. The H-index can be easily manipulated. arXiv:1304.2557 [cs.CC]

[10] Simon D Harding, Joanna L Sharman, Elena Faccenda, Chris Southan, Adam J Pawson, Sam Ireland, Alasdair JG Gray, Liam Bruce, Stephen PH Alexander, Stephen Anderton, et al. 2018. The IUPHAR/BPS Guide to PHARMACOLOGY in 2018: updates and expansion to encompass the new guide to IMMUNOPHARMACOLOGY. *Nucleic acids research* 46, D1 (2018), D1091–D1106.

[11] J. E. Hirsch. 2005. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences* 102, 46 (2005), 16569–16572.

[12] H. J. Imker. 2018. 25 Years of Molecular Biology Databases: A Study of Proliferation, Impact, and Maintenance. *Frontiers in Research Metrics and Analytics* 3 (2018). https://doi.org/10.3389/frma.2018.00018

[13] Jo McEntyre and Jim Ostell. 2002. The NCBI handbook. *Bethesda (MD): National Center for Biotechnology Information (US)* (2002).

[14] Chrystalla Pavlou and Edith Elkind. 2016. Manipulating citation indices in a social context. (2016).

[15] Eric W Sayers, Richa Agarwala, Evan E Bolton, J Rodney Brister, Kathi Canese, Karen Clark, Ryan Connor, Nicolas Fiorini, Kathryn Funk, Timothy Hefferon, et al. 2019. Database resources of the national center for biotechnology information. *Nucleic acids research* 47, Database issue (2019), D23.

[16] Conrad L Schoch, Stacy Ciufo, Mikhail Domrachev, Carol L Hotton, Sivakumar Kannan, Rogneda Khovanskaya, Detlef Leipe, Richard Mcveigh, Kathleen O'Neill, Barbara Robbertse, et al. 2020. NCBI Taxonomy: a comprehensive update on curation, resources and tools. *Database* 2020 (2020).

[17] J. A. Teixeira da Silva and J. Dobránszki. 2018. Multiple versions of the h-index: cautionary use for formal academic purposes. *Scientometrics* 115, 2 (2018), 1107–1113. https://doi.org/10.1007/s11192-018-2680-3

[18] René Van Bevern, Christian Komusiewicz, Rolf Niedermeier, Manuel Sorge, and Toby Walsh. 2016. H-index manipulation by merging articles: Models, theory, and experiments. *Artificial Intelligence* 240 (2016), 19–35.

[19] D. S. Wishart, Y. D. Feunang, A. C. Guo, E. J. Lo, A. Marcu, J. R. Grant, T. Sajed, D. Johnson, C. Li, Z. Sayeeda, N. Assempour, I. Iynkkaran, Y. Liu, A. Maciejewski, N. Gale, A. Wilson, L. Chin, R. Cummings, D. Le, A. Pon, C. Knox, and M. Wilson. 2018. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research* 46, Database-Issue (2018), D1074–D1082.

[20] Yinjun Wu, Abdussalam Alawini, Susan B Davidson, and Gianmaria Silvello. 2018. Data citation: giving credit where credit is due. In *Proceedings of the 2018 international conference on management of data*. 99–114.

[21] Alexander Yong et al. 2014. Critique of Hirsch's citation index: A combinatorial Fermi problem. *Notices of the AMS* 61, 9 (2014), 1040–1050.

Additional Material

A PROOFS

Proposition A.1. *Let A be an antichain in a ranked hierarchy H, and let $n \in A$ be a rank-minimal node of rank l. Then, there exists an l-antichain $A' \in H$ such that $|A'| \geq |A|$.*

Proof. Given $S_1, S_2 \subseteq H$, we say that $S_1 \sqsubseteq S_2$ if $\forall s_1 \in S_2$, $\exists s_2 \in S_2$ such that $s_1 \preccurlyeq s_2$, i.e., all the nodes in S_1 are also in S_2 or are descendants of a node in S_2. It is immediate to note that $\sqsubseteq$ is a partial order and, since H is a hierarchy, that if $S_1 \sqsubseteq S_2$ then $|S_1| \geq |S_2|$, since the nodes in S_2 cover the ones in S_1, they can be less than those in S_1.

Now let A be an antichain with a rank-minimal node n of rank l. If no node in A has a child with rank $\geq l$, then A is already an l-antichain. Otherwise, pick a node n' in A with at least one child with rank $\geq l$. Replace n' with all and only its children with rank $\geq l$ to obtain a new antichain A_1. We have that $A_1 \sqsubseteq A$, $A_1 \neq A$, and $|A_1| \geq |A|$. By repeating this process, we obtain a strictly decreasing sequence in $\sqsubseteq$, which must terminate in an l-antichain A' with $|A'| \geq |A|$. □

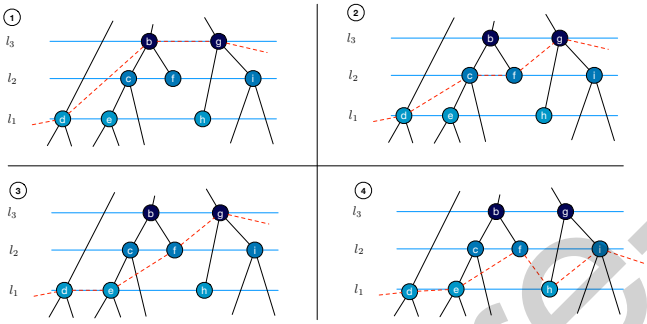

Figure 5: Example of the procedure described in Lemma 2.5 applied on one example hierarchy.

As an example of the procedure used in the proof of Prop. A.1, refer to Figure 5. At point 1, there is an l_1-antichain $\{d, b, g\}$. Since b and g present children with rank $\geq l_1$, it is possible to "expand downward" the antichain, including their children. At step 2, b is substituted with c and f. Continuing, at step 3, c is substituted with its child e, the only one at level l_1. At the last step, g is substituted with h and i. At this point there are no more children of the nodes in the l_1-antichain $\{d, e, f, h, i\}$ with children above the threshold l_1, and the process terminates.

B FURTHER DETAILS OF THE DATABASES

Drugbank. In this commercial database of drugs [2], the single drugs are the leaves in a taxonomy where the root is the entirety of the database, and the internal nodes are the classes of drugs. Drugbank includes a Linnean-style stratified classification scheme, as seen in the partial representation of Figure 1: Kingdom, Superclass, Class, and Subclass. In Drugbank, each drug is associated with a specific webpage with an URL like the following: https://go.drugbank.com/drugs/<id_of_the_drug>. We note that the papers referring to a Drugbank's drug often report its webpage identifier as a form of citation in the literature. Hence, using Google Scholar and the Drugbank ids of the drugs, we obtained the *number of citations* to each drug webpage. As indicate in the article, the number of citations collected with this method underestimates the actual number.

GtoPdb. As in Drugbank, the leaf nodes in the GtoPdb hierarchy correspond to individual substances. However it differs from Drugbank in two important respects. First, the hierarchy is not stratified into Linnean-style levels: there is a family/sub-family structure that varies in depth; currently the maximum length of a path is four. Second, in addition to the leaf nodes, the intermediate (family) nodes may also be cited. For example, both the leaf nodes and the intermediate nodes may carry a citable descriptive narrative. The hierarchy, which essentially underlies the web presentation of the database, [3] contains 4,079 nodes of which 3,286 are leaves.

The database has an accompanying *Concise Guide* [2] – a set of publications extracted automatically from the database every two years. Until recently, authors using the database were encouraged to cite the concise guide when they wanted to use some data in the database. Currently, every citable unit of the database is regularly published as an article in an online journal [6], which can cite and receive citations counted by citation systems like Google Scholar. In this work, we focus on the part of the database containing information about targets, since it is the part with an explicit hierarchical structure. In particular, by crawling the GtoPdb web pages concerning targets, it is possible to build this hierarchy following the hyperlinks in the pages. In GtoPdb each target (also known as receptor) is contained in a family. Each family, in turn, is contained in one of eight family types. The root of this hierarchy can be though as the part of the database dealing with targets.

Every two years GtoPdb publishes a data paper (e.g., [2]) summing up the content of the database. While these papers describe GtoPdb as a whole, other papers focus on one of the target family types, such as [1]. These can be considered as data papers *representing* the different parts of the database, in this case the family types.

Leveraging on this organization, we scraped Google Scholar getting the papers citing sections of above mentioned data papers about GtoPdb for the 2017-2020 timespan; hence, before the new publication/citation system was set up.

We then downloaded all (up to December 2020) the papers published in the British Pharmacological Society Journals[4] citing one or more of the GtoPdb data papers. We then divided these papers in terms of version of the database they cite (the 2017–2018 or 2019–2020 one) and in terms of which family type they cite. This means that the same paper, citing for example both Enzymes and Kinases, is classified in both categories. We were interested in the papers published specifically by the BPSJ since they must contain the URLs of the webpages they cited, following the editor's guidelines. It is therefore possible to scrape these papers looking from the citations to the webpages representing the single targets and target families, filtering these citations depending on the family

[2] https://go.drugbank.com/releases/latest

[3] https://www.guidetopharmacology.org/

[4] https://bpspubs.onlinelibrary.wiley.com/

type being targeted. This yielded the 7,870 citations to 1,446 of the 4,079 nodes in the hierarchy; note that 5,611 citations are to leaf nodes.

There is a second approach to obtaining the citations to GtoPdb, which is to count the incoming links to the target web pages. These links may come from conventional publications or from other web pages or data sets. Google analytics counts inbound links within a given domain and provides a list of 1,000 of these with the highest count. Unfortunately, many of the links in the GtoPdb domain are not to pages of the GtoPdb database. In fact, only about some 340 nodes in the database are counted with a total of approximately 75,000 links. This means that nodes with a low link count are not mentioned and the results are skewed. Because of the cut-off, the least (non-zero) cited node has a link count of 167. In fact, we did not consider them in our calculations in the article.

NCBI taxonomy. The National Center for Biotechnology Information[5] (NCBI) maintains what is surely the most comprehensive and best organized collection of medical data [13]. In particular it contains a catalog, Pubmed, of much of the world's biomedical literature and a number of databases. Through Entrez, an integrated retrieval system, links between 2.5 billion records are maintained [15]. This includes links to and from Pubmed references which could be treated as citations from conventional literature.

In the previous two examples (Drugbank and GtoPdb) our estimates of citation counts are low both because data citation is not widely practiced and also because it is often impossible thoroughly to "scrape" the literature for such citations. With NCBI and Pubmed, we believe that we have a much more realistic count of citations and links, and thus obtained an estimation of what a real link-based h-index would look like.

Unfortunately, using the API to obtain these link counts is slower than scraping them from the web interface. Therefore, we confined our attention to the subtrees of vertebrates and animals, which account for about half of the entire hierarchy.

[5] https://www.ncbi.nlm.nih.gov/