The Hitchhikers Guide to Sharing Graph Data

Matthew Roughan and Jonathan Tuke
ARC Centre of Excellence for Mathematical & Statistical Frontiers
School of Mathematical Sciences, University of Adelaide, Australia
Email: {matthew.roughan,simon.tuke}@adelaide.edu.au

***Abstract*—A graph is used to represent data in which the relationships between the objects in the data are at least as important as the objects themselves. Over the last two decades nearly a hundred file formats have been proposed or used to provide portable access to such data. This paper seeks to review these formats, and provide some insight to both reduce the ongoing creation of unnecessary formats, and guide the development of new formats where needed.**

***Keywords*-Graph, network, XML, GraphML, GML, database, data exchange, data representation.**

I. Introduction

Exchange of data is a basic requirement of open scientific research. Accurate exchange requires portable file formats, where portability means the ability to transfer (without extraordinary efforts) the data both between computers (hardware and operating system), and between software (different graph manipulation and analysis packages). The problem is exacerbated with big data, where human intervention cannot be expected to clean problematic datasets.

A search revealed that there are over 100 formats used for exchange of graph data. It seems that every new tool for working with graphs derives its own new format.

A small set of standardised formats for images (and other consumer data) is crucial for the functioning of digital society. Graph formats have affected a small community of sophisticated researchers and tool builders, so standardisation hasn't occurred because it just hasn't been that important. However, this community is growing, and the need for open exchange of information is growing, particularly where the data represent some real measurements that were expensive to collect.

The tendency to create new formats in preference to using existing tools is unhelpful, particularly as the time to "create" a format might be small, but the time to carefully test formats and read/write implementations is extensive. Reliable code is critical to maintain data quality, but many tool developers seem to focus on features instead of well-audited code. Moreover support of formats, for instance clear documentation and ongoing bug fixes, is often lacking.

An explosion of formats is therefore a poor state of affairs. The existing formats do include many of the features one might need, and some are quite extensible, so the bottleneck is not the features of the existing formats. We hypothesise that new formats are developed because detailed information about those already available is inaccessible. This is the gap that this paper aims to fill.

This work concentrates on graph *exchange* formats. Such formats have certain requirements above and beyond simple storage: most obviously portability. However, portability in this context is not purely about syntax. Exchange also requires common definitions of the meaning of the attributes. On the other hand, file size is not a primary consideration. Hence many exchange formats pay little attention to this and related details (e.g., read/write performance), but as exchange of very large datasets becomes important, the priorities may change.

Many of the formats presented may seem obsolete. Some are quite old (in computer science years). Some have clearly not survived beyond the needs of the authors' own pet project. However, we have listed as many as we could properly document, partially for historical reference, and partially to show the degree of reinvention in this area. But more importantly, because old and obscure isn't bad. For instance NetML, a format that doesn't seem to be used at all by any current toolkits, incorporates some of the most advanced ideas of any format presented. A good deal could be learnt if tool builders were to reread the documentation on this format.

It is important to note that this paper does not present yet-another format of our own. It is common in this and other domains for the discussion of previous works to be coloured by the need to justify the authors' own proposals. Here we aim to be unbiased by the need to motivate our own toolkit, and so (despite temptation) do not provide any such.

We do not argue that new graph formats should never be developed. In some applications new features are needed that are not present in the existing formats. However, it is critical that those who wish to propose new ideas should understand whether they are *really* needed. Moreover, in studying the existing formats, and their features, we learn what should be required in any new format to make it more than a one-shot, aimed at only one application. In fact, the results suggest that new formats are desirable for several reasons, but that perhaps what would be more useful would be a container format capable of providing self-documentation and meta-data-like features, while encapsulating a set of open encodings.

So the value of this work is threefold: firstly it provides a relatively complete set of information about the currently available formats, secondly it provides a basis for selection of a suitable format, and thirdly it provides information about the nature of the features that could be used in future developments of graph exchange strategies, in particular, the requirements of open, big data are that data formats supporting native compression become more universal.

II. Background

Graphs (alternatively called networks) have been used for many years to represent relationships between objects or people.

A mathematical *graph* $\mathcal{G}$ is a set of nodes (or vertices) $\mathcal{N}$ and edges (or links or arcs) $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$. The edges represent relationships between the nodes.

An alternative representation of a graph can be given through its *adjacency matrix* A, defined by

$$A_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases}$$

Other representations exist (and are discussed below in detail). These alternatives are often used to create computationally efficient operations on the graph.

Additional information is often added to a graph: *e.g.,*

- node or link labels (names, types, ...);
- values (distances, capacity, size, ...); and
- routing (paths taken when traversing the graph).

This additional information is often critical to make use of the graph data in any real application.

It has been necessary for many years for researchers in sociology, biology, chemistry, computer science, mathematics, statistics and other areas to be able to store graphs representing concepts as diverse as state-transition diagrams, computer-software structure, social networks, biochemical interactions, neural networks, Bayesian inference networks, genealogies, computer networks, and many more. Researchers also need to share data. They have done so by sharing files. As a result portable file formats for describing graphs have been around for decades.

This document is concerned with providing information about these formats, specifically with the intention of moving towards a smaller number of standard formats.

We only look here at publicly disclosed formats, for the obvious reason that a format can't really be called a data exchange format unless its definition is public. It is fair to say that although many were intended for exchange of information, most failed at this and were only really used for a single tool or database of graphs. In a few other cases, the format was not intended as an exchange format, but has become a *de facto* exchange format by virtue of the inclusion of IO routines in other software than its originator. In any case, we have tried to be inclusive here: we include anything that might be reasonably called an exchange format (and which is publicly documented to some degree), rather than trying to exclude those which we guess are not.

There are many subtypes of graphs, and generalisations. For instance: the general description above is that of a *directed* graph. An *undirected graph* has the property that if $(i,j) \in \mathcal{E}$ then so too is (j,i).

It is noteworthy that it is often possible to represent one type of graph in terms of the other: for instance an undirected graph may be represented by a directed graph by including all reverse links in the data. However, this is inefficient.

Moreover there is the issue of *intention*. The intention of the person storing the data is important: for instance, an undirected graph that is stored as a directed graph may be edited to become directed. A native undirected format enforces the correct semantics. Thus when considering the type of graph being stored, we consider only the explicitly supported subtypes.

Other generalisations of graphs include multi-graphs, hyper-graphs, and meta-graphs. Subtypes include trees and DAGs (Directed Acyclic Graphs). Once again, it is often possible to represent these in terms of the simple directed graph, but often this will be inefficient, and deficient in terms of intention. We will therefore look for native support for these generalisations and subtypes.

A. Related work

We distinguish this work from the study of *graph databases*, which have a similar role in storing data where the relationships have at least as much importance as the entities they relate. However, although they may hold the same type of data that we are considering here, the motivations for a graph database are different. Typically, those concerned with databases are interested in ACID (Atomicity, Consistency, Isolation, Durability) and other similar properties. The underlying assumption is that the data is changing dynamically according to some set of transactions and operations and that the database should work correctly under these conditions. Consequently graph databases are not simply concerned with the structure and description of the data, but also how that data may be operated on, and queried. On the other hand, the standard assumption in data exchange is that the data itself is relatively static, but portability is important.

There is a wide-ranging survey of graph databases [1], which is more concerned with the underlying database aspects, *e.g.,* the relationship between a graph database and other more traditional relational database, and the properties of various exemplar graph databases.

There is some overlap of concerns: in both cases there is some interest in data integrity, compression, and the like, but it is fair to say that these issues have taken second place in the design of graph exchange formats.

There have been a number of other efforts to gather similar information on graph exchange formats by researchers [2]–[4] and software distributors [5], [6]. The results provided inspiration for some of the descriptors used here, but this work aims at being more comprehensive. We surveyed approximately 100 possible formats, and include detailed data on 76, as compared to 7 in [3], 28 in [4], 5 in [5] and 11 in [6]. Bodlaj [2] considers a similar number of formats in general terms, but only 14 in detail, and with a different focus: for instance they provide more information on software support for formats than we provide here.

One additional paper to consider is [7], which was written specifically with the view of designing a new, more universal graph format. We deliberately avoid this approach in order to avoid bias in our discussion.

III. The File Formats

As noted the aim here is to describe graph *exchange* formats, *i.e.,* formats that are used to exchange data between scientists and programming environments. Not all of the formats started out that way – some were intended as internal formats for a particular software system, but have become *de facto* exchange formats when another system sought to leverage existing data by incorporating an existing format. A few of the formats are still primarily internal to a single system, but are important to describe because they exhibit an interesting feature. In the main we concentrate on those that were designed with data exchange in mind, or have been used in that way in practice.

This list is incomplete. There are some formats that we have observed in the literature, but are not adequately documented (*e.g.,* Gem2Ddraw), or which appear to only be used as an internal format for a single tool.

There are a few formats that we have lumped together under the general heading of *TGF* (the Trivial Graph Format) because they are all functionally equivalent to a delimited edge list. There is no point in listing every variant of this approach: there are many and they vary mainly on the choice of storage (plain ASCII through to Excel), and delimiter (tabs and commas are common).

There are many file formats that could, in principle, contain a graph: *e.g.,* XML, JSON, etc. For that matter any image file could contain an adjacency matrix. Unless there is a specific extension of these designed to provide support for graphs, in which case we list the specific not the generic.

We also aim to avoid, for simple practicality, formats that represent data that has a graph structure, but whose main content is not the graph. For instance HTML: the graph structure of the WWW is vastly smaller than the content and HTML is intended to store both in a distributed fashion.

Despite this focus, there were still 76 file formats considered. Consequently, we do not have space to present details of all of the file formats here. The details, and original data can be obtained from [8]. This paper provides summary statistics and analysis of the file formats.

We sought feedback on our data from authors or maintainers of the file formats, and received input for 23 of the formats. This may seem like a small proportion, but remember that many of these have not been maintained for some number of years.

In order to describe the formats we will consider here, we need some simple means to compare and contrast. Of a necessity, these will oversimplify some of the issues. What's more, many descriptions of file formats are imprecise. It is common to describe the format by reference to examples. Although useful for simple cases, these leave out important details: for instance: the character set supported, and even more surprisingly, the format of identifiers. It is often vaguely suggested that these are numbers, but without formal definition of what is allowed (presumably non-negative integers, but are numbers outside the 32-bit range supported?).

In the following, we make the best estimate of the capabilities of each format through reference to online documentation, and through a survey of the file format creators. In many cases the results are inferences, so in this section we will outline the features we describe, and the assumptions made in compiling our data. However, we have made the best effort possible to contact authors of formats, and their comments about capabilities have been given precedence.

Due to space limitations, we focus on some of the more interesting characteristics of the file formats. We provide a more complete analysis of features in [8], including details and references for all of the formats considered here.

A. Encoding

The primary encoding of the file is, in principle, a simple distinction in file type between text and binary files. However, text files today can use multiple different character sets, and this is important because some graphs will be labelled with non-English character sets. However, the majority of file-format definitions leave unspecified the character set to be used. We assume here that the character set is ASCII, unless there is some indication otherwise, either an explicit statement, or in the case of applications of XML it is assumed that the character set supported is Unicode. Figure 1 indicates the proportions of files providing each.

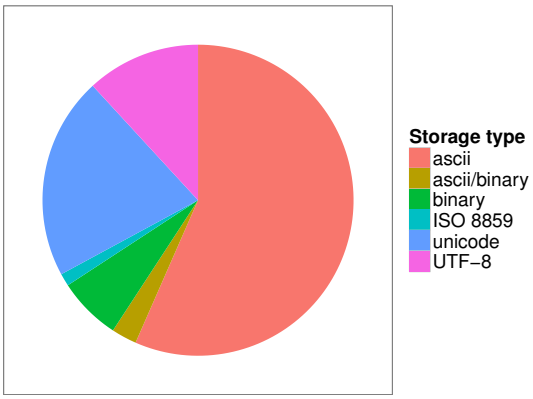

Fig. 1: File encoding proportions (total number is 76). It is noticeable that text-based encodings dominate. This reflects the need for portability in these formats.

B. Representation

The second critical decision in storing a graph is the choice of representation:

matrix : The graph's full adjacency matrix.
edge list : A list of the graph's edges [9].
smatrix : The matrix representation is poor for sparse graphs, which are common in real situations. However, some tools actually store a sparse matrix, which is almost equivalent to an edge list. There is a subtle difference in that a matrix view of the edges in a network cannot contain much detail about the edges.
neighbour lists : This is a list of the graph's nodes, each giving a list of neighbours for each node. Often called

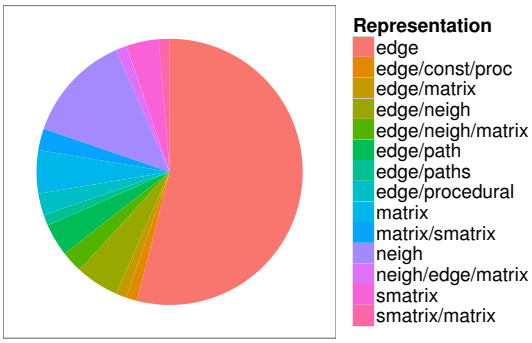

Fig. 2: Proportions of files supporting different representations (total number is 76). We list all the types of representation possible. Note that a substantial majority provide the edge list representation.

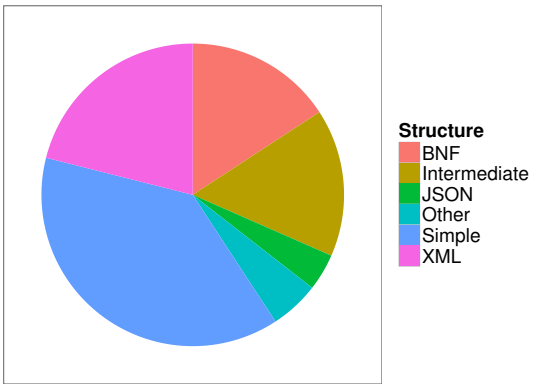

Fig. 3: File structure proportions (total number is 76).

adjacency lists we avoid that term because it is easily confused with the edge list.

path : One can also implicitly represent a graph as a series of *path* descriptions (essentially a path is a list of consecutive edges). This could be useful, for instance, with a tree or ring.

constructive : Graphs can often be described in terms of mathematical operations used to construct the graphs: for instance graph products on smaller graphs [10]. See [11] for a description of "levels" of graph formats.

procedural : Many graphs can be concisely defined by a set of procedures, rather than explicit definition of the nodes and links. This type of graph format could be very concise, but verges on creating another programming language. In fact, many graph libraries for particular programming languages essentially provide this, but in a non-portable manner.

These representations are given varying names in the literature, but we use the names above to be clear. Figure 2 shows the proportions of each type (note a file can support multiple representations).

The representation is important: for a graph with N vertices and E edges, the adjacency matrix requires $O(N^2)$ terms, the edge list $O(E)$ terms, and the neighbour list $O(N+E)$ terms. However, the terms in a matrix are $\{0,1\}$ whereas the terms in the edge and neighbour lists are node identifiers (consider they might be 64 bit integers), so the size of a resulting file based on each representation depends on many issues, including the way the data is stored in the file. No approach is universally superior.

Moreover, some may be easier to read and write: for instance a neighbour listing may be slightly more compact than an edge list, but the latter has the same number of elements per line, potentially making it easier to perform IO in some languages.

More subtly, a neighbour-list representation treats edges as properties of nodes, whereas an edge list treats edges as objects in their own right; and the matrix representation treats the graph as the only object with nodes and edges as properties of the graph. Although a program can internally represent data however it likes, and read in a neighbour list into structures that treat edges as objects in their own right, the native treatment of data is reflected in the ease with which attributes can be added. For instance, in a neighbour list it is intrinsically harder to record attributes for edges, and in the matrix representation it is harder to record attributes for nodes. This is, fundamentally, why we regard edge-list and sparse-matrix formats as different.

Some graph file formats allow alternative representations, and so we list all that are possible. However note that this is often actually multiple file formats under one name. It seems rare to allow a mixed representation.

We haven't (yet?) reported on whether edge-list formats explicitly lists nodes or only implicitly lists them as a consequence of edges. The latter is briefer, but requires a special case for degree 0 nodes.

When considering generalisations of graphs, other representations are possible (for instance tensors can generalise the concept of an adjacency matrix for multi-layer networks). However, codification of these is an ongoing research topic [12] and so we will not try to encapsulate it here.

C. File structure

This field describes how the file format's structure is defined. The cases are:

simple : the typical approach to create a graph format is to use one line per data item (a node, an edge, or a neighbourhood), with the components of a line separated by a standard delineator (a comma, tab, or whitespace). There are many variations on this theme, some more complex than others, for instance including labels, comments or other information. These formats are usually specified by a very brief description and one or two examples. They rarely specify details such as integer range or character set.

intermediate : this is a slight advance on a *simple* file format, in that it includes some grammatical elements.

For instance, the file may allow definition of new types of labels for objects. However, in common with simple files, these are usually only specified by a very brief description and one or two examples, not a complete grammar.

BNF : means that the file format is described using a grammar, loosely equivalent to a Backus-Naur Form (BNF). This is perhaps the most concise, precise description. When done properly it precisely spells out the details of the file in a relatively short form.

XML and JSON : many graph file formats extend XML, JSON, SGML, or similar generic, extensible file formats. This is a natural approach to the problem, and allows a specification as precise as BNF, though only through reference to the format being extended. Thus it is precise, but sometimes rather difficult to ascertain all of the details, unless one is an expert in XML, etc.

On the other hand, these approaches draw on the wealth of tools and knowledge about these data formats. On the other hand again, to use those tools the model of your graph object has to map to the XML model (or at least be easily transformed into that form).

Other : There are a small number that don't fit cleanly into a category: for instance procedural approaches.

Figure 3 shows the proportion of each type of structure within the files.

D. Evolution of Formats

We have included in our data a *reference time frame* to provide some historical context for the format. The dates are based on explicit records from the first recorded reference to the format, through to the last recorded date of maintenance. These should not be seen as a completely reliable data, but rather an attempt to document the historical development of this field, so much of which is not in the archival journals.

Figure 4 provides a quick summary of the evolution of the formats divided by file structure. The figure shows a smoothed distribution of the origin dates of formats, but structure. We can see that there was a flurry of activity in the late 90s continuing on until today, but the style of contributions has changed over time. It is interesting to see how XML became flavour of the year around in the late 90s, and then dropped out of popularity in recent years, and in the most recent past there seem to be several efforts to design graph formats on top of JSON. It seems there are fads even within technical fields.

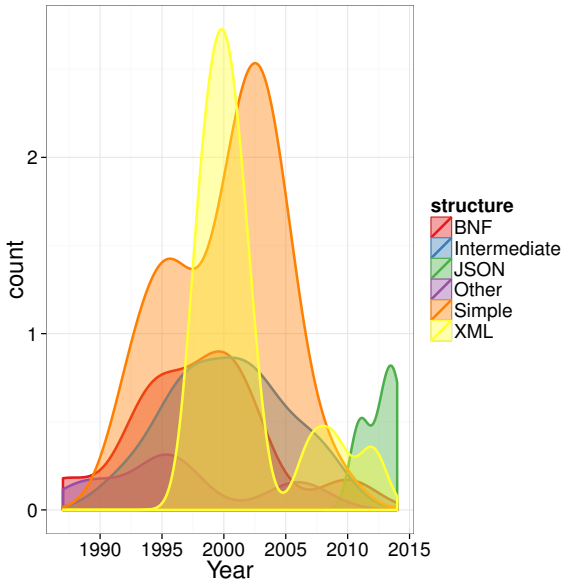

Fig. 4: New format origination dates (count gives the averaged number per year for a type of format).

IV. File Features

In considering the file formats we examined some 27 potential features of the format. We describe here the most interesting.

integral meta-data : Meta-data is data about the graph: for example, its name, its author, the date created, and so on. This is very important data, but many formats provide no means to include it in the file, and instead rely on external records. We refer to meta-data as *integral* if it is contained in the file itself.

Some formats allow meta-data through unstructured comments. This is better than nothing, but lack of structure of the comments means these are not machine readable, in general.

Some file formats provide only a limited range of meta-data fields, whereas others are arbitrarily extensible. Figure 5 shows support for various types of meta-data in the formats.

The value of meta-data is clear, but once again, let us reiterate that there are pros and cons in different approaches. For instance, arbitrary meta-data may seem superior, but can then lead to ambiguity about what meta-data should be kept for each dataset, whereas having a fixed list of attributes can make it obvious what is expected. However, it is common for formats to have support downwards, *e.g.,* formats with fixed attributes often also support comments, and those with arbitrary properties often support some set of fixed properties and comments.

built-in compression : It is easy enough to compress a graph-file using common utilities such as gzip, and typical compression ratio will be reasonably good as graph files often have many repeated strings. However, one format (BVGraph) provides for compression of the graph as it is written, in much the way image file formats allow intrinsic compression of the image.

Graph Compression algorithms have been a topic of study at least since 2001 [13]–[15], with numerous follow-ups. So it is interesting that only one format is designed around this feature. However, two other formats provided some crude mechanisms to reduce the size of the file. Finally the DGS (the Dynamic GraphStream) format [16] formally acknowledges the role of compression by requiring

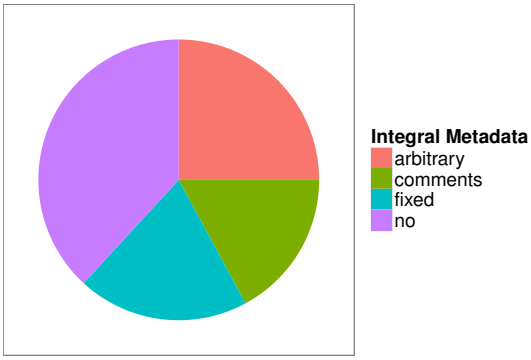

Fig. 5: Support for meta-data.

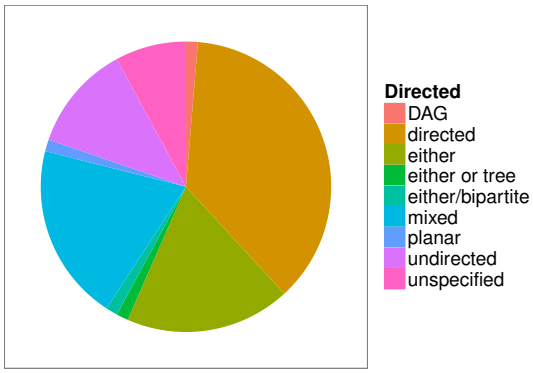

Fig. 6: Graph type support.

that a gzipped file be accepted by software reading its format.

graph types supported : As earlier noted there are many generalisations and subtypes of graphs that could be supported here: directed vs undirected, multi-graphs, hyper-graphs, meta-graphs, trees, DAGs, etc. Figure 6 shows the complexities of the issues involved by level of support just for directed graphs and their subtypes (either refers to either directed or undirected, and mixed allows a mixture of directed and undirected edges).

hierarchy : It is common for graphs to have sub-structure, for instance nodes that themselves contain graphs.
Several formats provide mechanisms to record this sub-structure. Unfortunately, there does not seem to be a consistently used definition of this type of structure [17], and so we see differences not just in the representation, but also what exactly is being represented. The problem becomes even more complicated when hierarchy and hyper-graphs are combined [12] (there is at least one proposed solution [17] but it does not seem to be widely used yet).
Here, we simply note whether the format provides a version of hierarchy.

attribute support : A very common requirement is to store sets of values associated with an edge, or node. We describe the level of support as arbitrary (where arbitrarily named attributes can be added) or fixed (where only a set of pre-defined attributed are allowed). Figure 7 shows the proportions that allow each type. Some formats also allow a single numerical weight to be attached to an edge (see Figure 8 for the proportion).

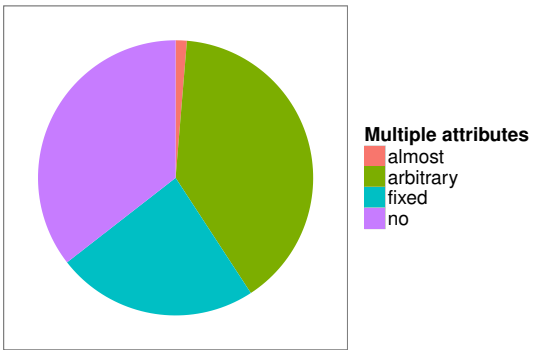

Fig. 7: Attribute support.

temporal data/dynamics : A topic of interest is analysis/visualisation of graphs as they change [9]. One way to store this information is as a series of "snap-shot" graphs, but storing it all together in the same file has some appeal. A few formats provide some variant on this: allowing links or nodes to be given a lifetime, or proving "edits" to the graph at specific epochs.

extensible : Some formats allow extensibility in varying forms. We only consider them to have this facility, however, if they provide an explicit mechanism. For instance, we do not regard all XML derivatives as intrinsically extensible because they could, in principle, be extended using standard XML techniques. The format has to explain the explicit mechanism whereby it is extended.
Simply adding extra attributes is not considered extensibility.

schema checking : A format that provides an explicit mechanism to check that a file is in a valid format is useful. We only say it has this facility if a tool exists to perform the check (a schema-checking program, DTD, or other similar formal tool).

checksums : It is possible for large data files to become corrupted. A common preventative (or at least check for this problem) is to use a checksum. This is possible for all files, but we say that a given format has this capability if it includes it as a internal component (usually checking everything except the checksum itself). Only a few formats contain this check.

multiple graphs : Some formats allow multiple graphs to be held in one file. Again, we only count this as a feature if the specification explains how explicitly.

To explore, we look at each level and calculate the proportion of formats supporting the features. This is plotted in

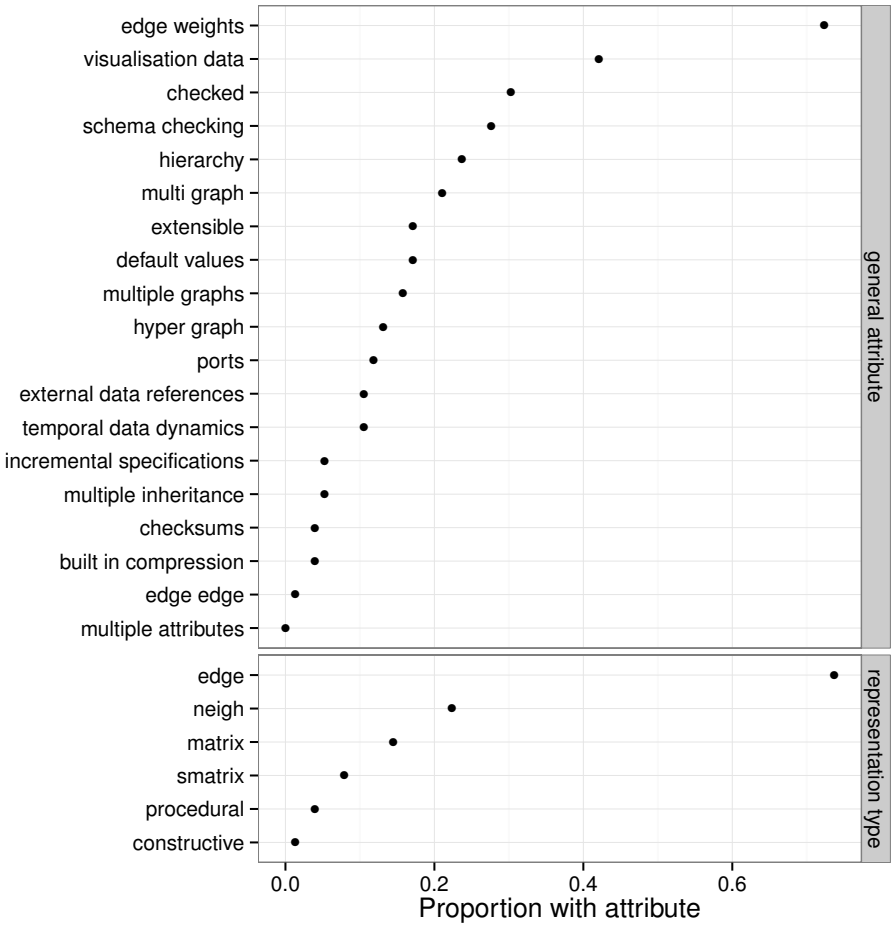

Fig. 8: Proportion of each feature in the data formats.

Figure 8. Note that in regard to features with multiple answers (*e.g.,* representation), we break the possibilities into categories and list the proportion that support each category.

V. Other Issues

A. Software Support

An important issue in choosing an exchange format is how many software tools support the format. The list of potential software is long, even more so than the list of formats, so we won't try to survey them here as well. Instead we refer readers to [2], which contains a cross-section of both formats and their software support.

A common conclusion amongst those who look at graph file support in software is that GraphML and Pajek are the best supported in modern systems, but they are by no means universal or even supported by the majority of tools.

B. Public DB support

The other type of support we might wish to see is general support amongst those who provide data publicly. There are many public databases that provide example networks for benchmarking or research. We provide a list in [8] of some of the better known of these with their format choices. Additional data sources are listed in [18], and a detailed taxonomy and examples of computer-network data appears in [19].

There is no clear winner here: slightly preferred is a variant of the Trivial Graph Format due to its least-common-denominator status (but note that this isn't really one format, so much as a collection of equivalent formats). Overall, however, the formats seem to be written for the data rather than the other way around. That, in itself, is an illustration of the problem.

C. Discussion

We said at the start that we would distinguish this study from that of the graph databases, and that steers the discussion towards features related to portability. However, the consideration of these formats inevitably leads to the question of what is currently missing. Apart from support of new graph generalisations and other features discussed above, the main features that are missing are those that might reasonably come from the database field. For instance, at present the vast majority graph exchange formats assume that the data will be read into memory in its entirety. As datasets become larger this may not be possible, and so exchange formats that support more advanced features such as subdivision of the data, or random access queries on the data, may be preferable.

VI. Conclusion

The science of graphs and networks needs portable, well-documented, precisely-defined, exchange formats. There are many existing formats, and this paper seeks to unravel this mess, most notably with the aim of reducing the number of new formats developed.

One size probably does not fit all though. There is a clear need for at least three major types of file format:

- a general, flexible, extensible approach such as GraphML;
- a quick and dirty approach that satisfies the least common denominator for the exchange of information to/from the simplest software; and
- a very efficient (compressed) format for very large graphs.

In the context of big data, the last type of format is the most interesting, The only true example of a compressive format is BVGraph does not allow attributes, so this points to a new area of research, namely the design of formats that work in conjunction with graph compression algorithms.

Acknowledgements

This work was supported by ARC grant DP110103505, and by the ARC Centre of Excellence for Mathematical & Statistical Frontiers.

Many people have contributed to improve the quality of information presented here; specific thanks go to Andreas Winter, Andy Schürr, Brendan McKay, Danny Bickson, Ivan Herman, Kevin Kawkins, Mason Porter, Michael Himsolt, Peter Mucha, Rok Sosic, Sebastian Mueller, Skye Bender-deMoll, Syd Bauman, Sébastien Heymann, Tels, Ulrik Brandes, Vladimir Batagelj, Bruce Hendrickson, David Gleich, Young Hyun, Antoine Dutot, Rose Oughtred, the BioGRID Administration Team, and David Krackhardt.

References

[1] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, pp. 1:1–1:39, Feb. 2008, http://dl.acm.org/citation.cfm?id=1322433. [Online]. Available: http://doi.acm.org/10.1145/1322432.1322433

[2] J. Bodlaj, "Network data file formats," 2013, http://link.springer.com/referenceworkentry/10.1007/978-1-4614-6170-8_298.

[3] M. Bernard, "Graph file formats," www2.sta.uwi.edu/~mbernard/research_files/fileformats.pdf.

[4] S. Bender-deMoll, "Netwiki: Data formats for representing networks," 2010, http://netwiki.amath.unc.edu/DataFormats/Formats.

[5] "yWorks Developer's Guide: input and output," 2014, http://docs.yworks.com/yfiles/doc/developers-guide/io.html.

[6] "Gephi: Supported graph formats," http://gephi.github.io/users/supported-graph-formats/.

[7] U. Brandes, M. S. Marshall, and S. C. North, "Graph data format workshop report," in *8th International Symposium on Graph Drawing*, 2000, pp. 410–418.

[8] M. Roughan and J. Tuke, "Unravelling graph-exchange file formats," March 2015, http://arxiv.org/abs/1503.02781.

[9] J. Ebert, "A versatile data structure for edge-oriented graph algorithms," *Computing Practices*, vol. 30, no. 6, pp. 513–519, 1987.

[10] E. Parsonage, H. X. Nguyen, R. Bowden, S. Knight, N. J. Falkner, and M. Roughan, "Generalized graph products for network design and analysis," in *19th IEEE International Conference on Network Protocols (ICNP)*, Vancouver, CA, October 2011.

[11] V. Batagelj and A. Mrvar, "Towards NetML: Networks markup language," in *International Social Network Conference*, London, July 1995, vlado.fmf.uni-lj.si/pub/networks/netml/snetml.pdf.

[12] M. Kivelä, A. Areanas, M. Barthemlemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *Journal of Complex Networks*, vol. 2, pp. 203–271, 2014.

[13] M. Adler and M. Mitzenmacher, "Towards compressing web graphs," in *Data Compression Conference (DCC)*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 203–212. [Online]. Available: http://dl.acm.org/citation.cfm?id=882454.875027

[14] K. H. Randall, R. Stata, J. L. Wiener, and R. G. Wickremesinghe, "The link database: Fast access to graphs of the Web," in *Data Compression Conference (DCC)*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 122–131. [Online]. Available: http://dl.acm.org/citation.cfm?id=882455.874988

[15] P. Boldi and S. Vigna, "The WebGraph framework I: Compression techniques," in *Thirteenth World-Wide Web Conference*, 2004, pp. 595–601.

[16] "The DGS file format specification," http://graphstream-project.org/doc/Advanced-Concepts/The-DGS-File-Format_1.1/.

[17] D. Bildhauer and J. Ebert, "DHHTGraphs - modeling beyond plain graphs," in *IEEE ICDE Workshop*, 2011, pp. 100–105, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5767620&tag=1.

[18] M. Cerinsek and V. Batagelj, "Sources of network data," in *Encyclopedia of Social Network Analysis and Mining*, R. Alhajj and J. Rokne, Eds. Springer New York, 2014, pp. 1946–1954, http://link.springer.com/referenceworkentry/10.1007%2F978-1-4614-6170-8_313. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-6170-8_313

[19] G. D. Battista and M. Rimondini, *Handbook of Graph Drawing and Visualization*. CRC Press, June 2013, ch. Computer Networks, pp. 763–803, http://cs.brown.edu/~rt/gdhandbook/.