The Compound Graph: a case study for Community Visualisation in Social Networks

Chris Walshaw
Computing & Mathematical Sciences
University of Greenwich, London, UK
c.walshaw@gre.ac.uk

***Abstract*— This paper builds on previous work which aimed at providing a graph-based visual exploration of melodic relationships (tune families) within collections of traditional music. Here, using a community detection algorithm, potential tune families can be readily identified. However, the richer the information contained in the graph, the more difficult it is for the visualisation algorithms to operate successfully. Therefore, an approach is proposed which uses modified versions of the graph both to enhance the community detection results and, more importantly, restructure the graph, by creating a *compound graph*, to reveal the communities visually. Finally, the wider applicability of the technique is considered.**

Keywords—melodic similarity, network analysis, graph drawing

I. Introduction

This paper is a follow-on to the ideas presented at the previous Conference on Information Visualization, [1], the aim of which was to conduct a visual exploration of melodic relationships within collections of traditional melodies. There, by representing the relationships using a graph which was then visualised it was possible to identify "tune families" (strongly clustered groups of vertices). However, limitations of the previous work were the observational approach to the results and limits to the scalability and applicability of the techniques. Whilst those limits still exist, in this paper ideas are presented which both extend the robustness and scalability of the approach.

The common idea is that, given a collection of tunes and a melodic similarity measure which can compute pairwise similarity between tunes (e.g. [2]), it is possible to construct a complete proximity graph of the corpus. In this corpus graph each vertex represents a tune and edge weights represent similarities between tunes: the greater the similarity the larger the edge weight. If a similarity threshold is applied so that an edge is only included in the graph if the two tunes it connects are sufficiently similar then a sparse proximity graph can be induced (the higher the threshold, the sparser the graph).

Next a community detection algorithm (not used in [1]) can be applied to determine possible tune families and finally a graph layout algorithm can be used to visualise the resulting graph with the communities superimposed.

In the previous paper, [1], although many tune families were readily identifiable visually, a significant limitation was that the richer in information (and therefore denser) the graphs become, the more difficult it is to visualise them. Here a technique is proposed which uses the results of the community detection algorithm to manipulate/simplify the graph. This allows the visualisation of far denser graphs than previously.

In the following, Section II outlines the three main components of the procedure (A) graph construction; (B) community detection; and (C) graph layout. Section III then discusses the graph manipulations, including the construction of the compound graph, that aid the process. Section IV presents the results and extends the idea to benchmark graphs (unrelated to music collections). Finally section V concludes the paper.

II. Methodology

A. Proximity graph: multilevel recursive alignment

To construct the graph, the melodic similarity measure used is multilevel recursive sub-sequence alignment. Space precludes a full description but it is discussed in [1] & [3].

The multilevel aspect refers to the fact that each tune is coarsened by recursively removing non-stressed notes to create a hierarchy of tune representations. When two tunes are compared, the similarity is calculated at each level of the hierarchy and then aggregated across the levels. This means that tunes which may differ in minor details can still be considered sufficiently similar, in terms of stressed notes (i.e. at a coarser levels), to generate edges in the corpus graph. It also allows for significant improvements in computational cost.

This multilevel similarity measure, S(X,Y), induces a complete weighted graph on a dataset, where the edge weight between each pair of melodies is given by the similarity. Subsequently, when the graphs are displayed, edge thickness can be shown in proportion to the weight with very similar vertices joined by thick edges and not so similar ones by thin edges.

Since most edges in the graph will have very small weights it makes sense to restrict the graph to include only edges for tunes which are reasonably close matches. This restricted graph is referred to henceforth as the **corpus graph**. It can be created in a variety of ways but here it is assessed by a **matching threshold**, T, and edges between melodies X and Y are only included if they match across at least some proportion T of their length. Specifically an edge between vertices V_x and V_y is only included if

$$S(X,Y) \geq \text{average}(\text{length}(X), \text{length}(Y)) * T$$

Following [3] values for T in the experiments are 1/4, 1/6 and 1/8: the former value considerably restricts the number of edges since it requires tunes to match across at least a quarter of their length (in fact, typically it is significantly more than a quarter because of biasing towards longer aligned sub-sequences, [1]); the latter threshold is fairly inclusive but can allow many false positive matches.

B. Community detection

Algorithms for detecting clusters, or **communities**, of well-connected vertices within a graph are an important topic in network science and social network analysis. Although there are no universally accepted procedures and even the definition of a community is not well-defined, there has been considerable progress in recent years and a comprehensive survey of the topic can be found in Fortunato & Hric, [4].

Here a popular community detection algorithm due to Blondel *et al.* is used, [5]. Coincidentally, but satisfyingly in a paper where both melodic similarity and graph drawing use the multilevel paradigm, Blondel's method is also multilevel in design.

The algorithm is a heuristic which aims to optimise the **modularity** of the clustering, where modularity is a scalar value between −1 and 1 that measures the density of internal community links as compared to links between communities, [5]. The algorithm is initialised by adding every vertex to its own community (so that every vertex is in a community of 1) and then iterating repeatedly through the vertices, moving them to another community if that move increases the modularity.

When no further improvements can be made (i.e. the clustering has reached a local maximum of the modularity function), the vertices in each cluster are contracted to form a single vertex in a coarsened version of the graph and then the modularity optimisation is applied to this coarsened graph. This procedure is then repeated recursively until a coarsened graph is found where no improvement to the modularity is possible.

C. Visualisation: multilevel force-directed placement

Having constructed the connected corpus graph, it can be visualised using multilevel force-directed placement (MLFDP), a heuristic method for drawing graphs which uses a multilevel framework combined with an FDP algorithm. FDP is a common technique for computing a layout of graph vertices, e.g. [6], which models edges as springs (that push / pull vertices apart) and includes global repulsive forces to untangle the graph.

However, FDP is unable to untangle large-scale structures and so is limited to graphs of a few hundred vertices. Hence multilevel schemes were introduced both to accelerate the computation and, more importantly, impart a global quality to the drawing, [7]. The multilevel technique matches and coalesces pairs of adjacent vertices to define a new graph and is repeated recursively to create a hierarchy of increasingly coarse graphs. The coarsest graph is then given an initial random layout and the layout is refined iteratively with FDP and recursively extended to all the graphs starting with the coarsest and ending with the original. Since its introduction, the multilevel approach has become widespread as a framework for improving graph drawing algorithms, e.g. see [6] and the references therein.

III. Graph manipulations

So far the work combines three (multilevel) technologies: melodic similarity to build the proximity graph, community detection to identify strongly connected clusters of vertices (putative tune families) and force-directed placement to visualise the graph. However, as the results will indicate, there are still significant problems in viewing the denser graphs.

Although density is adjustable using the T parameter (section II.A) there is a trade-off: if T is too large, the graph becomes very sparse and disconnected (with a high proportion of isolated vertices). Whilst easy to visualise, typically the community detection indicates far more communities than the tune families annotated manually. Furthermore, the sparsity misses out on much information about weaker relationships between melodies. Conversely, if T is reduced, the graph becomes much richer in information, but also very dense, making it impossible to visualise successfully (e.g. see the "hairball" in Fig. 3(left)). Moreover, reducing T too far does not seem to add any useful information: it just increases the number of false positive edges.

To address these issues, three manipulations are performed: graph connection, edge scaling and use of the compound graph.

A. Graph connection

As will be seen, most of the graphs produced for the results section comprise several **disconnected components** (subgraphs that are not connected by any edges) and often large numbers of **isolated vertices** (vertices with no incident edges – i.e. tunes that are not similar to any other tune in the corpus). This presents a problem for force-directed graph drawing techniques as, without adapting either the algorithm or the graph, the components will be repulsed far apart from each other (as attractive forces only operate along edges).

Here, therefore, a simple scheme is proposed to connect the graph (less computationally complex than that described in [1]):

1. The corpus graph G(V, E) is constructed as discussed, with edges in E only included between pairs of vertices if the corresponding tunes are sufficiently similar.
2. The connected components are evaluated and sorted in order of increasing size.
3. Starting with the smallest component, all potential edges from that component to other components are calculated, and the edge with the heaviest weight is added in to the graph, but with its weight set to zero.
4. Repeat from step 2 until the graph is fully connected.

It is easy to see that this scheme adds nothing to the total edge weight in the graph, but increases the number of edges by |D| – 1 where |D| is the number of disconnected components in the original proximity graph.

The purpose of working from smallest to largest component is so that the graph doesn't (necessarily) agglomerate around what is the largest component initially. The purpose of picking the heaviest edge is to focus on the strongest connections (even though these edges will be relatively light as they have already been excluded by the proximity graph threshold, T).

Setting zero edge weights is important for visualisation: since edge weights influence vertex placement, a zero-weight edge will have minimal impact on the graph layout but will mean that the two insufficiently similar vertices that it connects are positioned as close together as possible. Furthermore zero-weight edges can be easily excluded in the visualisations. Experiments were also tried with non-zero edge weights but actually made the Variation of Information scores (see section IV.B) much worse.

B. Edge scaling

Initial results indicated that reducing the threshold T to give a richer, more expressive graph made the community detection harder: when T is reduced from one value to another the additional edges included should be lightly weighted (indicating weaker relationships), but the sheer number of additional edges overwhelms the community structure encapsulated in the heavier edges. In the worst cases, for very small values of T, the community detection algorithm was unable to find any structures and returned the whole graph as the only community.

Another way to view this is that, if there is an optimal value of the threshold, T* say, then additional edges included for values of $T < T^*$ are likely to be inter-community edges rather that intra-community. Therefore, weakening those additional edges should help retain the community structure found for T*.

This is the basis of the edge scaling scheme, although rather than weakening a set of edges introduced when T is reduced from some unknown value T*, the procedure simply uses a step function to reduce the weight of all edges beneath another threshold, W, say. Thus if an edge weight is $\leq W$, its weight is reduced to 1 for the community detection phase. (A number of other scaling schemes were also tested but the step function works well and is simple to implement.)

Then, in order to choose an optimal value for W, the method tests all values of W between 1 (i.e. no scaling) and some upper limit W_L (set at 50 in the experiments below) by running the community detection algorithm repeatedly and choosing W* as the value of W which results in the highest modularity.

C. The Compound Graph

The third and probably most valuable graph manipulation technique is the use of a compound graph to aid with the visualisation. Except for the larger values of T, many of the graphs calculated for the results are not possible to visualise using standard force directed placement techniques as they are far too dense. The solution is to simplify/summarise some of the information contained in the graph and the community structure gives a means for doing that.

To motivate this, ultimately the aim is to provide a semi-automated visual tool to help identify "families" of closely related tunes. In the case of annotated datasets with known tune families (ground truth), although difficult to match those families exactly, it is possible to pick values of T where either:

(1) [for larger values of T] the number of automatically detected communities exceeds the number of tune families and so for datasets without ground truth it might be a case of manually identifying and combining separate communities which are in fact part of the same family;

(2) [for smaller values of T] the number of known families typically exceeds the number of automatically detected communities and so for datasets without ground truth the researcher's task is to split communities into families.

In the former case, it may be helpful to look at the overall strength of relationships between communities (rather than pairwise relationships between vertices); in the latter case it could be argued that inter-community edges are a distraction.

The suggestion then, is to use the community structure to illuminate the information presented in the graph. The use of community structures has a long history in graph visualisation, e.g. [8], where each community is laid out separately before being composed to complete the layout, or [9], where community structures are used to inform edge sparsification and thus render visualisation tractable. Here the idea is to bundle together all the edges between pairs of communities and use additional vertices, one per community, as a link between inter-community and intra-community edges. The approach is related to techniques in [6], where edges are bundled together (using splines, unlike here) and [10], where vertices of the **quotient graph**, G′, are attached to those of the original graph, G.

The quotient graph is the result of applying a partition or clustering, P, to the original G (so that $G' \approx G/P$). Each cluster of vertices in G is represented as a single **super-vertex** in G′ and edges between two clusters are merged into a single edge between the super-vertices representing those clusters. Edge weights are summed so the total edge weight in the cluster graph is $||E_C||$, where $\{E_C\}$ is the set of inter-cluster edges (or edges cut by the clustering). This is essentially the same procedure as creating a coarsened graph in the multilevel schemes.

To merge the original graph and quotient graph the simplest way is to remove all the inter-community edges from the original graph (now summarised in the quotient graph). Then an extra vertex-to-cluster edge is added between every original vertex and the super-vertex representing its cluster. Henceforth this merged graph will be known as the **compound graph**, $G'' \approx G + G/P = G(1+1/P)$.

It is easy to see that the compound graph has $|V| + |C|$ vertices, where $|V|$ is the number of vertices in the original and $|C|$ is the number of communities. The number of edges decreases by $|E_C|$, but also increases by $|V|$, as there is one edge per vertex connecting the vertex to its cluster, and by $|E'|$, where $|E'|$ is the number of edges in the quotient graph. This gives a total of $|V| + |E'| - |E_C|$ additional edges (where the total may often be negative).

Total vertex and edge weight depend on how the super-vertices and vertex-to-cluster edges are weighted. Typically in the quotient graph each super-vertex is given the total weight of the clustered vertices it represents (so $||G'||$, the total weight of vertices in G′, is the same as $||G||$). However, when the compound graph is visualised (using FDP) this has the effect of adding a large weight at the centre of each cluster and repulsive forces hollow out the cluster with original vertices pushed away in a ring. However, this is easy to deal with by simply giving each super-vertex zero weight. Thus, the total vertex weight of the compound graph, $||G''||$ is the same as $||G||$ and $||G'||$.

Slightly trickier are the edge weights on vertex-to-cluster edges. If they are set to a small value (e.g. 1) then the effect is that the heavily weighted inter-cluster edges (which may combine many original edges) dominate in the FDP drawing of the compound graph: typically, all the super-vertices are pulled into a tight structure in the centre with clusters of original vertices pushed out to the periphery. Instead, it is preferable that each super-vertex is tightly bound to the vertices in its cluster and that the inter-cluster edges are then used to place the clusters relative to each other.

To achieve this in the context of FDP each vertex-to-cluster edge needs a relatively heavy weight, k, say. In fact, the melodic similarity measure has an artificial but hard limit of 256 (so that any melody matching with another by more than 256 consecutive notes is limited to a similarity score of 256). Therefore, every vertex-to-cluster edge is given a weight of k = 256 and this seems to have the desired effect. Thus, the total weight of edges in G″ is given by $||E|| + k\,|V|$ where $||E||$ is the total weight of edges in the original graph. In the experiments k = 256 but, provided it is large, the actual value is probably not crucial (k = 128 also worked well).

IV. RESULTS

A. Datasets

The previous paper used two small, manually annotated datasets of around 360 vertices each to validate the ideas. Since then two further large datasets with ground truth have come to light – one is a recently published and much larger version of one of the original datasets; the other is a web-based collection called TheSession.org – a large and growing collection of ~30,000 Irish traditional tunes hosted at a community website.

An important realisation about TheSession is that when members publish a tune and that tune is a variant of one that already exists on the website, the tune is published *on the same page* as the first one. Assuming that members don't inadvertently miss an existing tune and don't publish together variants that are not really related, this means that each page which contains more than one variant identifies the tunes on that page as a ready-made, crowd-sourced tune family!

Furthermore, to get smaller collections which also have ground truth, one simply has to take an ordered subset of tunes (e.g. the first 5,000) from TheSession and the coherence of the tune families is maintained. (The same is not true of the other collections, where such an operation would be possible but more involved).

In all six collections were tested in eight datasets (3 small with ~360 tunes, 2 medium with ~5,000, and 3 large with ~10-12,000 tunes). The collections, including those with no ground truth (i.e. no known tune families), are as follows:

- **MTC**: The Annotated Corpus of the Meertens Tune Collection[1], version MTC-ANN-2.0.1, [11]. **360 tunes in 26 families**.
- **Morris**: English morris dance tunes from the Morris Ring website[2], [1]. **368 tunes in 111 families**.
- **TS-***: TheSession.org[3]. Three different sized subsets were used for comparison purposes:
  - **TS-360: 360 tunes in 177 families**
  - **TS-5k: 5,000 tunes in 2,837 families**
  - **TS-10k: 10,000 tunes in 6,017 families**
- **VMP**: The Village Music Project[4], a website containing English dance music from C18th/C19th manuscripts. Currently the site hosts ~7,000 tunes but here a benchmark subset, [1], is used. **5,638 tunes, no ground truth**.
- **MTC2**: The Meertens Folk Song Collection[1], version MTC-FS-INST-2.0, [12]. This contains 18,618 melodies with 12,762 annotated as belonging to 5,297 tune families. Neglecting the ~6,000 unannotated tunes and a few where the multilevel parsing was not possible (see [3] for limitations) this leaves **12,322 tunes in 5,092 families**.
- **Essen**: An abc version of the Essen Folk Song Database[5], containing European and Chinese folk songs. **10,186 tunes, no ground truth**.

Table I shows the results for all eight datasets using matching threshold values, T = 1/4, 1/6 and 1/8. The table shows |E|, the number of edges; |D|, the number of disconnected components; |C|, the number of communities detected; Q, the optimal modularity; and VI, the variation of information score. In datasets with ground truth the VI indicates how closely the communities match the families, with 0 being an exact match, so the best (smallest) values are highlighted (see below for details).

Characteristics for the compound graphs, G″, are not shown but can be partially derived. Thus the number of vertices in the compound graph, |V″|, is given by |V| + |C| (the |V| values are not in the table but are listed above).

TABLE I. DATASET CHARACTERISTICS

Dataset	T	|E|	|D|	|C|	Q	VI
MTC	1/4	556	148	152	0.925	0.219
MTC	1/6	2,079	30	52	0.898	**0.179**
MTC	1/8	16,311	2	18	0.594	0.280
Morris	1/4	200	267	269	0.960	0.196
Morris	1/6	518	153	163	0.958	**0.150**
Morris	1/8	2,518	24	49	0.844	0.330
TS-360	1/4	166	246	246	0.947	0.072
TS-360	1/6	260	217	217	0.934	**0.056**
TS-360	1/8	388	172	178	0.937	0.100
TS-5K	1/4	1,812	3,744	3,744	0.996	0.049
TS-5K	1/6	3,239	3,135	3,149	0.996	**0.044**
TS-5K	1/8	13,816	1,702	1,879	0.972	0.219
VMP	1/4	3,156	4,187	4,187	0.991	n/a
VMP	1/6	5,895	3,225	3,287	0.990	n/a
VMP	1/8	69,425	914	1,102	0.867	n/a
TS-10k	1/4	3,511	7,565	7,567	0.998	**0.042**
TS-10k	1/6	6,979	6,224	6,264	0.997	0.058
TS-10k	1/8	57,828	2,980	3,314	0.939	0.279
MTC2	1/4	14,085	7,100	7,161	0.994	**0.131**
MTC2	1/6	318,682	2,461	2,819	0.790	0.395
MTC2	1/8	5,373,920	387	518	0.379	0.681
Essen	1/4	11,470	6,338	6,411	0.960	n/a
Essen	1/6	746,865	1,180	1,317	0.479	n/a
Essen	1/8	8,649,146	175	199	0.367	n/a

B. Metrics

The question of how to evaluate the techniques is not an easy one to resolve. However, for collections where tune families are already known (as a ground truth), one obvious answer is to

[1] http://www.liederenbank.nl/mtc/
[2] https://themorrisring.org/music/handbook-morris-dances
[3] https://thesession.org/
[4] http://www.village-music-project.org.uk/
[5] https://ifdo.ca/~seymour/runabc/esac/esacdatabase.html

compare the communities identified with the known families to see how well they match. Even this is not straightforward as there is no universally agreed measure which can compare the similarity of two clusterings (partitions), [4]. Nonetheless in that paper Fortunato & Hric recommend the Variation of Information (VI) metric proposed by Meilă, [13], and that is the partition similarity measure used above. Note the values in Table I have all been scaled to lie between 0 and 1 (where 0 indicates a perfect match) but VI is not directly comparable for different size graphs as the maximum value is log |V|, where |V| is the number of vertices in the graph.

For collections with no ground truth another option might be to use the maximum modularity, Q, found by the community detection algorithm, for example to choose an appropriate value for T. However, this is not necessarily helpful; see for example the two compound graphs shown in Fig. 2, with T values set to 1/4 and 1/6 respectively. The left hand one looks to correspond to case (1) in section III.C – communities will need to be merged into families, whilst the right hand one may be case (2), the converse. However, for both the modularity is almost identical, 0.990 as compared with 0.991. So it may simply be a case of trying different values of T to produce a suitable graph.

C. Discussion

Table I shows how rapidly the connectivity grows as T is reduced (as exemplified by |E| and |D|). None of the graphs listed is fully connected (|D| = 1), but several are close. Unsurprisingly the number of communities detected, |C|, is closely correlated with the number of components, |D|, particularly for the TS-* datasets, but they can differ widely, especially when T = 1/8. Likewise the modularity, Q, is very high in very disconnected graphs, but can drop off rapidly as T is reduced. Nonetheless, the best values of VI, i.e. the best matches with ground truth, do not always match the highest modularity values.

In terms of VI, the optimal values consistently occur for T = 1/6 for the small and medium sized graphs and for T = 1/4 for the larger graphs. Indeed they are startlingly good for the TS-* datasets although this may just indicate that the crowd-sourced tune families are quite conservative. In other words, it is likely that members of the website only publish very closely related tune variants on the same page (this would also help to explain the high correlation of |D| with |C|).

Space precludes any detailed discussion of the edge scaling results (section III.B) but the technique doesn't always have a dramatic effect (for example when the modularity is already high, T = 1/4). However, as an illustration of its effectiveness, for the Essen dataset with T = 1/6 (Fig. 3), using a value of W* = 23 increased the modularity from 0.279 to 0.479 and the number of communities detected from 1,239 to 1,317.

Turning to the visualisations, benefits of the compound graphs are readily apparent. Fig. 1(left) & 3(left) both show the original corpus graph with no manipulations (other than being connected). The communities are superimposed but are still difficult to decipher fully for the small MTC dataset (Fig.1) and impossible for the Essen dataset (Fig. 3). However, when the compound graph is constructed and is then used to drive the layout calculated by the MLFDP algorithm, the communities are easily visible, Fig. 1(right) & Fig. 3(right), even though they may need work to manually separate out larger communities.

Fig. 2, meanwhile, shows compound graphs for the medium sized VMP dataset at two different values of T, which most likely illustrate the two cases described in section **Error! Reference source not found.**

D. Wider applicability of the compound graph

The original intention of this paper was to discuss the construction and, in particular, visualisation of proximity graphs derived from melodic similarity measures. Typically these graphs are very sparse with many disconnected components. However, when tried with denser graphs (Fig 3.) the compound graph was found to be very helpful in visualising the underlying structure. Accordingly the technique was tried on a number of benchmark graphs taken from the literature.

A full evaluation goes well beyond the scope of this paper, but two sample pictures are shown in Fig. 4. On the left is a compound graph visualisation of the Gnutella06 peer-to-peer network[6], with 8,717 vertices and 31,525 edges, [14]; on the right is the Smith60[7] network from the Facebook100 dataset with 2,970 vertices and 97,133 edges, [15] (a visualisation of this graph also appears in [9], Fig. 11).

In both cases MLFDP visualisations of the original graph (not shown) produce "hairballs", similar but more compressed, to that in Fig 3(left). However, the compound graphs, Fig. 4, clearly able to separate out the communities and elucidate the structure. Obviously the communities themselves can be further explored by zooming in and even at this level some intra-community structure is evident.

Note that in these pictures the thick inter-cluster edges are rendered as semi-transparent, so as not to obscure details of the communities they originate from. Also, and as above, the vertex-to-super-vertex edges are weighted with $k = 256$. In fact the visualisation seems to be relatively insensitive to this parameter – it just needs to be large enough to keep super-vertices at the centre of their cluster. However, these edges are not shown in any of the Figures as they do not add any useful information.

The communities in these cases were detected using Blondel's modularity optimisation algorithm (section II.B) but without edge scaling (section III.B) as the original graph edges are not weighted. However, in principle any community detection method could be used.

Of course, this compound graph visualisation is very dependent on the success of the community detection algorithm; if no communities are detected the compound graph is meaningless and changes to the detected community structure could dramatically change the visualisation. Nonetheless, it seems a promising technique and could even be used hierarchically, in the same way that community detection algorithms are sometimes employed.

[6] https://snap.stanford.edu/data/p2p-Gnutella06.html

[7] http://networkrepository.com/socfb-Smith60.php

V. Conclusions

This paper uses a community detection algorithm both to help identify tune families and to aid with the visualisation of those families. Although it does not seem possible to automatically calculate these families, the technique presented here does seem to offer a useful and (in principle) interactive tool to music researchers.

Perhaps more importantly, the results indicate that not only can the visualisation expose communities (as previously), but also that using the community structure to drive the visualisation can significantly enhance the visibility of the communities. In particular, the use of the compound graph, can allow exploration of previously un-visualisable graphs – potentially an interesting technique with much wider applications.

In that respect it should be stressed that the ideas discussed here are generic, both in terms of the case study (in principle, any pairwise melodic similarity measure could be used) and also in terms of the graph drawing problem: the technique as a whole can be applied to any proximity graph, the edge scaling to any graph with weighted edges and the compound graph construction to any graph where communities are detectable.

Acknowledgments

The author would like to thank Doron Goldfarb for suggestions leading to the development of this paper and the authors of Gephi, Open Graph Viz Platform[8], for their software.

References

[1] C. Walshaw, "A Visual Exploration of Melodic Relationships within Traditional Music Collections," in *22nd Intl Conf. Information Visualisation*, 2018, pp. 478–483.

[2] B. Janssen, P. van Kranenburg, and A. Volk, "Finding occurrences of melodic segments in folk songs employing symbolic similarity measures," *J. New Music Res.*, p. (to appear), 2017.

[3] C. Walshaw, "Constructing Proximity Graphs To Explore Similarities in Large-Scale Melodic Datasets," in *6th Intl Workshop on Folk Music Analysis*, 2016.

[4] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Phys. Rep.*, vol. 659, pp. 1–44, 2016.

[5] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech.*, vol. 10, p. P10008, 2008.

[6] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger, "Multilevel Agglomerative Edge Bundling for Visualizing Large Graphs," in *IEEE Pacific Visualization Symposium*, 2011, pp. 187–194.

[7] C. Walshaw, "A multilevel algorithm for force-directed graph-drawing," *J. Graph Algorithms Appl.*, vol. 7, no. 3, 2003.

[8] X. Wang and I. Miyamoto, "Generating customized layouts," in *Graph Drawing*, 1996, pp. 504–515.

[9] A. Nocaj, M. Ortmann, and U. Brandes, "Adaptive Disentanglement Based on Local Clustering in Small-World Network Visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 6, pp. 1662–1671, 2016.

[10] C. Walshaw, "Variable partition inertia: graph repartitioning and load-balancing for adaptive meshes," in *Advanced Computational Infrastructures for Parallel and Distributed Adaptive Applications*, S. Chandra and et al, Eds. Wiley, New York, 2010, pp. 357–380.

[11] P. van Kranenburg, B. Janssen, and A. Volk, "The Meertens Tune Collections : The Annotated Corpus (MTC-ANN) Versions 1.1 and 2.0.1," 2016.

[12] P. van Kranenburg and M. de Bruin, "The Meertens Tune Collections: MTC-FS-INST 2.0," 2019.

[13] M. Meila, "Comparing clusterings — an information based distance," *J. Multivar. Res.*, vol. 98, no. 5, pp. 873–895, 2007.

[14] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford Large Network Dataset Collection," 2014. [Online]. Available: http://snap.stanford.edu/data.

[15] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of Facebook networks," *Phys. A Stat. Mech. its Appl.*, vol. 391, no. 16, pp. 4165–4180, 2012.

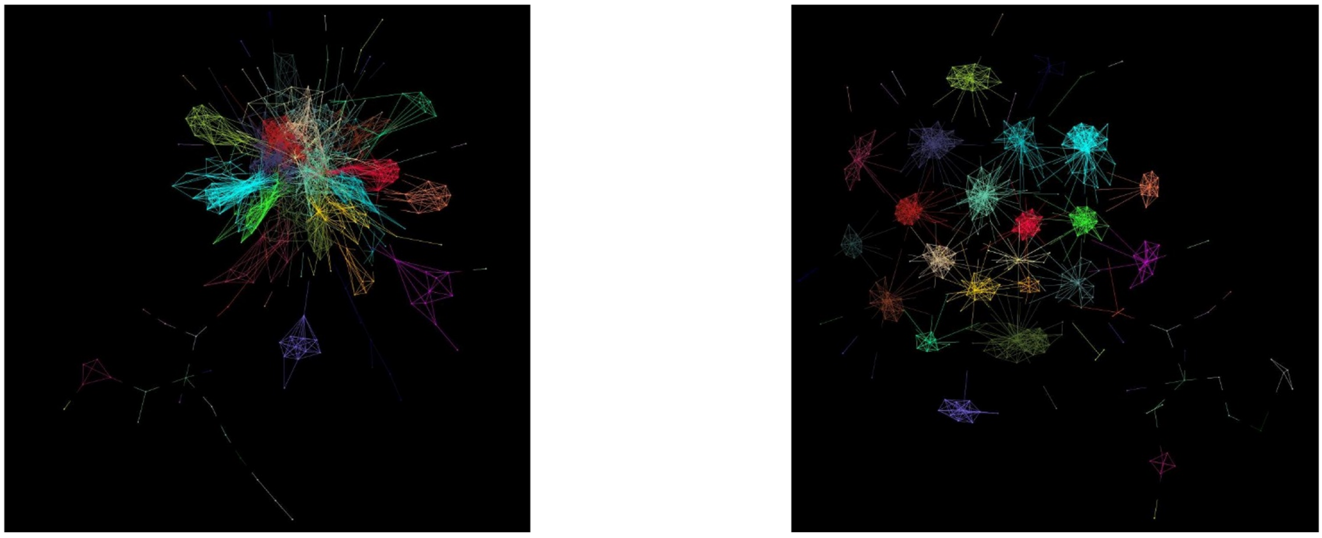

Fig. 1. Small corpus graphs for the MTC collection with T = 1/6, showing the original graph (left) & compound graph (right)

[8] https://gephi.org/

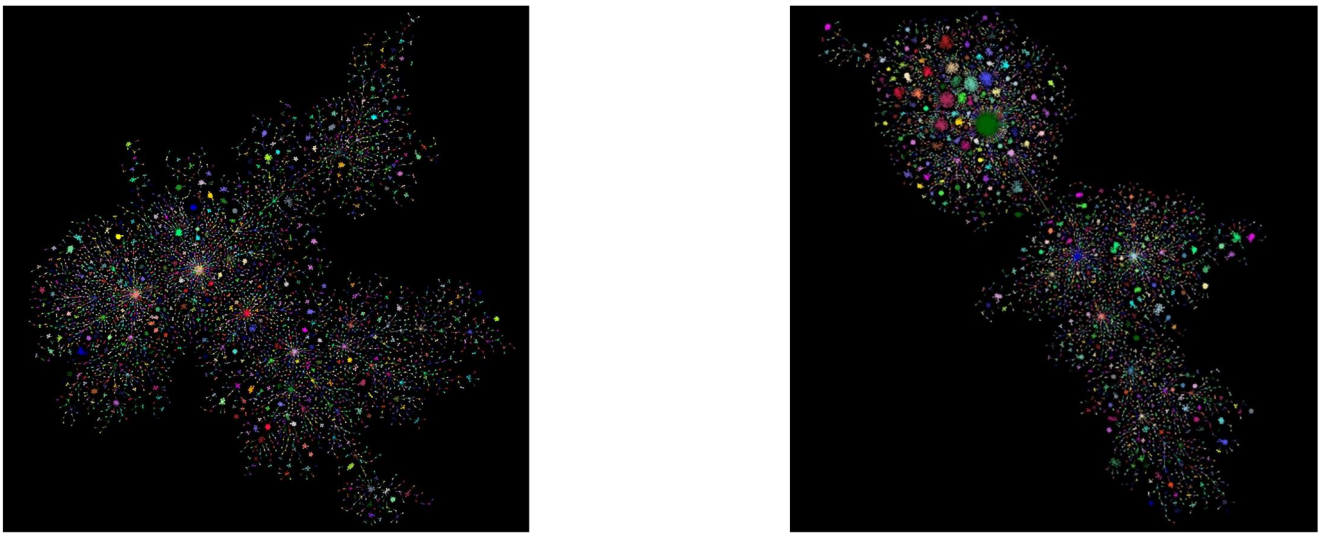

Fig. 2. Medium sized compound corpus graphs for the VMP dataset with T = 1/4 (left) and T = 1/6 (right)

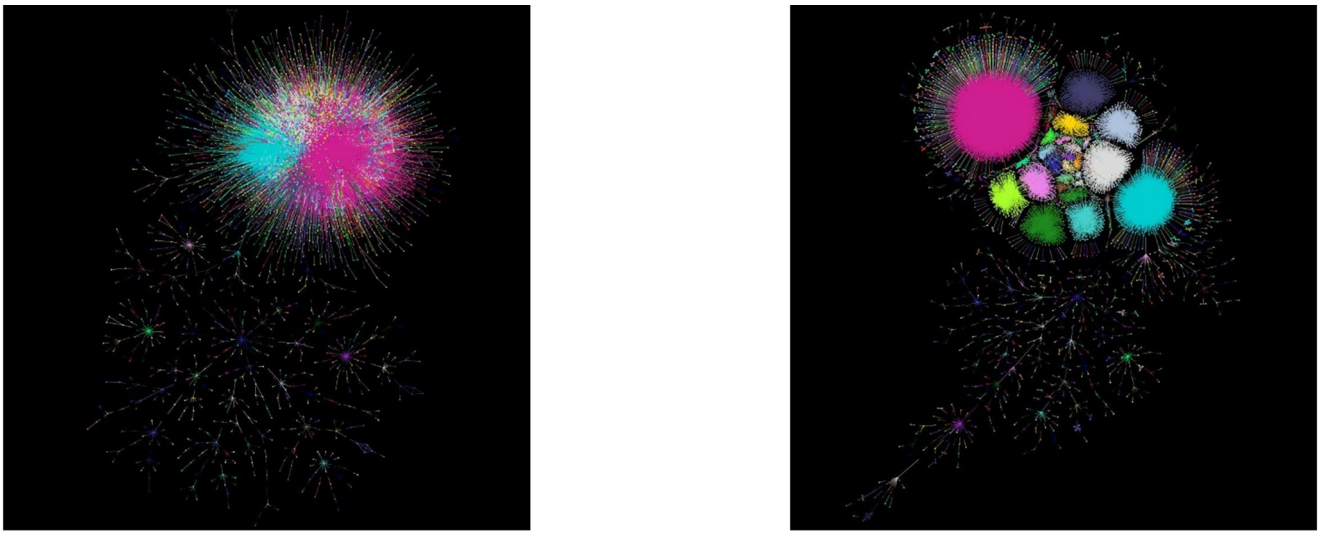

Fig. 3. Large corpus graphs for the Essen collection with T = 1/6, showing the original graph (left) & compound graph (right)

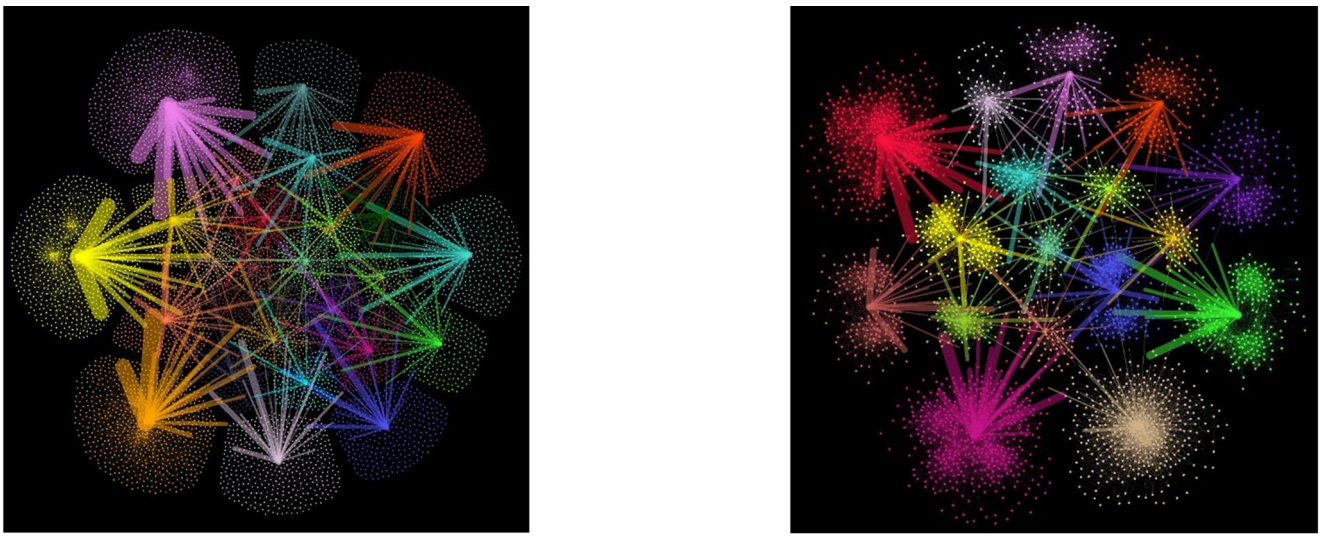

Fig. 4. Compound graphs for benchmark networks taken from the literature, showing Gnutella06 (left) and Smith60 (right)