
Proceedings Paper:
Gosliga, J. orcid.org/0000-0003-3997-3224, Bunce, A., Hester, D. et al. (1 more author) (2021) Creating a network of structures based on physical similarity. In: Cunha, A. and Caetano, E., (eds.) Proceedings of the International Conference on Structural Health Monitoring of Intelligent Infrastructure. SHMII-10 : 10th International Conference on Structural Health Monitoring of Intelligent Infrastructure, 30 Jun - 02 Jul 2021, Porto, Portugal (online). International Society for Structural Health Monitoring of Intelligent Infrastructure (ISHMII) , pp. 1803-1808.

Creating a network of structures based on physical similarity

Dr Julian Gosliga[1], Andrew Bunce[2], Dr David Hester[2], Prof. Keith Worden[1]

Dynamics Research Group, Dept of Mechanical Engineering, University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK[1]
1Civil Engineering, School of Natural and Built Environment, Queens University Belfast, Stranmillis Road, Belfast, BT9 5AG, Northern Ireland[2]

email: j.gosliga@sheffield.ac.uk, abunce01@qub.ac.uk, d.hester@qub.ac.uk, k.worden@sheffield.ac.uk

ABSTRACT:
Effective structural health monitoring (SHM) requires large amounts of data representing the normal condition of a structure as well as any damage conditions. However, it is not always feasible to obtain these data; for example, it is not economical to obtain damage-state data for a new bridge. To address this problem, a new framework is being explored called population-based structural health monitoring (PBSHM), which proposes that if two structures are sufficiently similar, then data can be shared between them.
Tools which enable the sharing of data, such as the transfer of models and damage classifiers, have been explored in previous work; as have methods for assessing the similarity of structures.
This paper will describe how it may be possible to link structures based on their physical similarity in such a way that creates a network, with communitities of similar structures. Within these communities, data can be shared between structures.
Forming these communities in a way that is computationally efficient while still avoiding missing possible links is not straightforward. This paper outlines some of the considerations that must be taken into account for solving this problem.

KEY WORDS: population-based structural health monitoring, irreducible element model, networks, communities

1 INTRODUCTION

Previous foundational work [1–3] on population-based structural health monitoring (PBSHM) has described the benefits of taking such an approach to SHM, namely the ability to share data and models between structures that can be considered sufficiently similar, as well as providing a robust normal condition and improved damage detection in *homogeneous* populations [4, 5]. For *strongly-homogeneous* populations, a single model can be used to describe the generic structure behaviour and the characteristic variations within the population [1]. For *heterogeneous* populations, transfer learning methods can be employed to improve the performance of classifiers by leveraging information gained from examining similar structures [3]. Determining whether or not two structures are similar enough for any of these data transfer methods to be applicable is achieved through the use of *irreducible element* (IE) models [2].

The IE model for a structure will eventually live within a database, where similarity comparisons are performed automatically, based on the SHM problem specified. The IE model encodes information that is important for determining the physical similarity of two structures: geometry, material properties, and boundary conditions.

Similarity comparisons between IE models are actually performed on the *attributed graph* of the structure. Attributed-graphs have been used before in mechanical engineering to represent purely geometrical data, with the view to estimate the machining costs for various components [6, 7]. In the current case, the attributed graph contains the same information as the IE model, but arranged into a different form which makes it less human-readable, but which makes the data more accessible to comparison algorithms.

These comparison algorithms take the attributed graphs and seek to return some measure of their similarity. There are numerous methods for comparing graphs, such as methods based on spectral graph theory [8], which compare the spectra of the Laplacian matrix for two different graphs. Other methods find the size of the maximum common subgraph (MCS) between two graphs [9,10] and others use graph neural networks to generate vector embeddings of a graph and then return the distance between the resulting vectors [11]. In some cases the comparison is based purely on the topology of the graph (spectral graph comparison [8] and MCS without attributes [9]), and in others both the topology and attributes are taken into account (graph neural networks [11] and MCS with attributes [10]).

In most cases where network theory is applied, for example examining the internet or social networks, the network is already defined by pre-existing connections between the objects in question. Forming a network of structures, on the other hand, is a different problem as there needs to be some criteria for when connections are formed and when they are not. It is this question for which this paper provides a brief exploration. Not only is there the question of which comparisons to perform when adding a new structure to the network, but also which comparison method is most suitable? There is a constant balance to be struck between making meaningful connections between structures which provide useful information and computational cost. Sparse networks may miss important connections between structures, while a network in which every comparison possible is performed may be needlessly expensive to run.

2 COMPUTATIONAL COMPLEXITY OF DIFFERENT APPROACHES FOR GENERATING NETWORKS

In this section, two approaches which represent the extremes of how to determine which comparisons to perform will be described and their relative merits discussed.

The first approach is to compare any new structure that is added to the network to every other existing structure in the network. The communities are then formed using a clustering algorithm such as k-NN, which will group structures that are closest (most similar).

The second approach compares the new structure to one other existing structure chosen at random. If the distance is below a certain threshold, the new structure is added to the community that the existing structure belongs to, if not another existing structure is chosen, and so on until eventually a sufficiently close structure is found. If no such structure exists, then the structure forms a new community.

The third approach compares any new structure with a randomly-selected structure from each of the existing communities within a network. The structure is then added to a community if the distance is below a certain threshold. In the case that the new structure is sufficiently close to more than one community, the structure is added to the closest community. The structure is then compared to every other structure within that community.

The first approach performs the maximum number of comparisons possible, but is guaranteed to always produce the same communities, whereas the second is unpredictable in the number of comparisons possible, although it would be expected to be less than the full number of comparisons possible. The third approach strikes a middle ground between the two.

2.1 Fully-connected approach

One possible approach for creating the network involves calculating the pair-wise distance between any new structure S_{new} and all the existing structures within the network $S_1, ..., S_n$. This approach will be referred to as the 'fully-connected approach'.

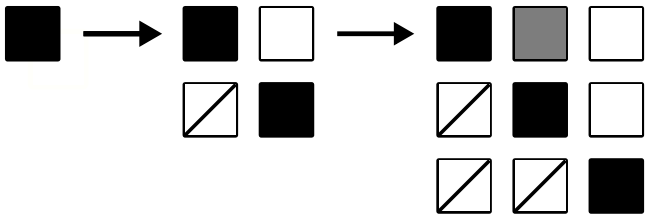

Figure 1: The image represents the entries in the distance matrix for the network as new structures are added. An arrow represents a new structure (and hence new row and column) being added to the network. A black square indicates that the distance is between the structure and itself, which should always be 0. A grey square indicates that the distance has been calculated in a previous step. The lower triangle does not need to be calculated if using a metric since the distance matrix will be symmetric.

Using the fully-connected approach, finding communities within the network becomes a clustering problem.

Figure 1 shows the distance calculations necessary for generating the fully-connected network. Examining this figure, it becomes clear that the complexity of adding a new structure to the network using this approach is $O(n-1)$.

The complexity of matching graphs G_1 and G_2 using the Bron-Kerbosch algorithm [12] is $O(3^{\frac{|V|}{3}})$, where $|V| = |V_1| \cdot |V_2|$. Exact graph comparison has a high computational cost associated with it and should be avoided where possible. The computational cost for performing comparison operations remains the same whichever approach is chosen; however, certain approaches may require less comparisons to be performed.

The number of edges in a fully-connected network is $|E| = \frac{n(n-1)}{2}$. The number of edges can be reduced by clustering using a k-NN algorithm and removing any edges between structures that fall outside of that cluster. Using a k-NN clustering algorithm would lead to the following number of edges, $|E| = \frac{kn}{2}$.

2.2 Demonstration of the fully-connected approach

Figure 2 shows a fully-connected network generated by performing pairwise comparisons between eight different bridges, where some are the same type; for example, two beam and slab bridges, two truss bridges, etc. In Figure 2, bridges with similar abbreviations are the same type, B&S1 and B&S2 are both beam and slab bridges. The thicker lines represent structures that appear closer together.

To form communities, it is necessary to use a clustering algorithm such as k-NN. Here the number of nearest neighbours dramatically affects the communities that are formed. The same structures as in Figure 2 are now clustered using a k-NN algorithm where $k = 3$ (Figure 3) and $k = 2$ (Figure 4).

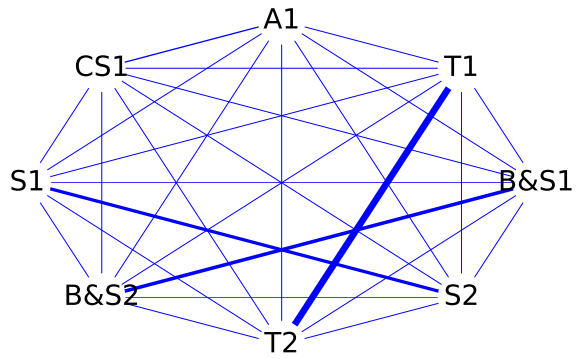

Figure 2: The fully-connected network formed between eight different bridges of various types. B&S stands for beam and slab, T stands for truss, A stands for arch, CS for cable-stayed, and S is for suspension bridge. The thicker the line, the closer the two structures.

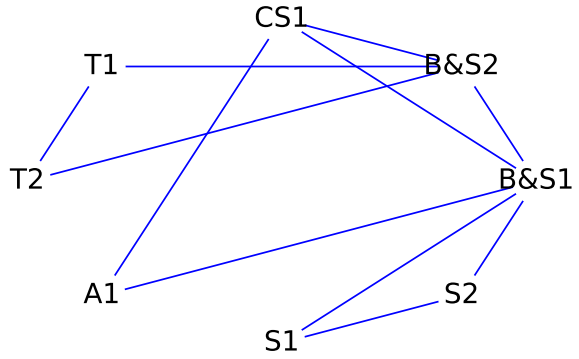

Figure 3: Clustering performed by applying k-NN to the network shown in Figure 2, where $k = 3$. B&S stands for beam and slab, T stands for truss, A stands for arch, CS for cable-stayed, and S is for suspension bridge.

With $k = 3$, it is not clear where the communities lie exactly, while $k = 2$ gives distinct communities which contain bridges of the same type. It is likely that a more intelligent clustering algorithm is required, as the size of communities will not be constant. Some communities may have many members and others very few. Also, the size of the communities will change over time.

2.3 Community comparison approach

Another possible approach, which would lead to a reduced number of comparisons, would be to compare any new structure to one structure chosen at random from each community that exists in the network. This ensures that only one structure from each community is compared to the new structure. If the new structure is sufficiently close (distance below a certain threshold) to one of the structures in an existing community, it then becomes a member of this community and is compared to every member of that community. In the case that the structure is sufficiently close to randomly selected structures from multiple communities, it is added to the closest community. If the new structure falls outside the threshold to be added to any of the existing communities, it forms a new community.

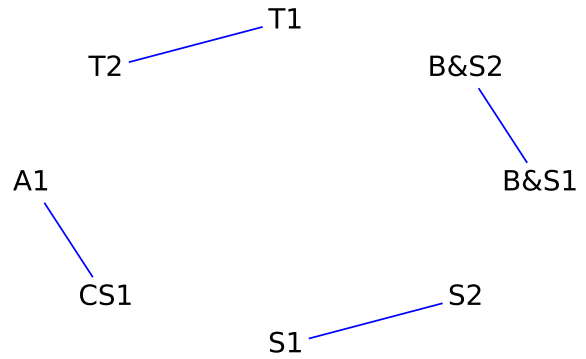

Figure 4: Clustering performed by applying k-NN to the network shown in Figure 2, where $k = 2$. B&S stands for beam and slab, T stands for truss, A stands for arch, CS for cable-stayed, and S is for suspension bridge.

S_{new} is compared to a random structure from a community $\mathbf{S} \in C_1$. If the distance $d(S_{new}, \mathbf{S}) < \epsilon$ (where ϵ is some threshold), and $\mathbf{S}$ is indeed the structure closest to S_{new}, then $S_{new} \in C_1$. This is done for all communities $C_1, ..., C_m$, where m is the number of communities. This step therefore requires m comparisons.

The next step is to compare S_{new} to all structures $S_2, ..., S_i \in C$, where C is the community to which S_{new} was assigned in the previous step. (Assuming that S_{new} was compared to S_1 when assigning the community label.) Since all the communities are likely to vary in size, it is more useful to look at the average number of comparisons performed in this step. On average, $i = \frac{n}{m}$, which then means the number of comparisons required is $\frac{n}{m} - 1$.

The average complexity of this approach is then $O(m + (\frac{n}{m} - 1))$.

The number of edges in the resulting graph is determined by the fact that the network is fully connected within the community and every node has $m - 1$ connections to nodes in other communities.

If on average, each community has $\frac{n}{m}$ nodes, the total number of edges within each community will be $\frac{n(\frac{n}{m}-1)}{2m}$. Across the whole network this becomes $\frac{n(\frac{n}{m}-1)}{2}$. (As $m \to n$, the number of edges within communities goes to zero.)

Each node within the community also has $\frac{m-1}{2}$ edges associated with it, so for each community, there are $\frac{n}{m}\frac{m-1}{2}$ edges that span across communities, giving $\frac{n(m-1)}{2}$ for the whole network.

Approach	Complexity	Edges
Fully-connected (FC)	$O(d(n-1))$	$\frac{n(n-1)}{2}$
FC with k-NN	$O(d(n-1)+kn)$	$\frac{kn}{2}$
Community	$O(d(m+(\frac{n}{m}-1)))$	$\frac{n}{2}(m+\frac{n}{m}-2)$

Table 1: Table comparing the computational complexity of adding a new structure to the network, and number of edges in the resulting network for each approach. Here, d is the cost of calculating the distance between a pair of graphs, n is the number of structures currently in the network, m is the number of communities in the network, and k is the number of neighbours selected for k-NN.

This gives the total number of edges in the entire network as $|E| = \frac{n(\frac{n}{m}-1)}{2} + \frac{n(m-1)}{2} = \frac{n}{2}(m+\frac{n}{m}-2)$. If $m = 1$, $|E| = \frac{n(n-1)}{2}$, the number of edges in a fully-connected network.

If there was only one community in the network, the amount of comparisons required for the first step would be 1, and the next step would require the new structure to be compared to every other structure in this community and hence the network. This is a case that may occur early on when populating the network if only similar structures are being added. In this case, the computational complexity is the same as the fully-connected method.

2.4 Seeding communities

While it is possible to start with a blank network and add structures one at a time, a better approach might be to 'seed' the network with IE models that represent each of the classes of structures that are of interest (bridge, aeroplane, wind turbine). The representative IE models for each class of structure would capture what could be considered *essential* features of that particular class; for example, a bridge requires a deck at the very least, and an aeroplane should feature wings and a fuselage. This way structures are less likely to be added to the wrong community, since they should all be close to the example for their particular class of structure.

2.5 Comparison of approaches

A summary of the computational complexity and the number of edges in the resulting network can be found in Table 1. While these are two obvious ways of quantifying the difference between various approaches, they do not capture the whole picture and included below are further points to consider when examining the three approaches.

Points to consider for the fully-connected approach:

- Guaranteed to produce identical results for same set of structures.
- Fully-connected graphs have a high edge density and it is likely to contain a lot of unnecessary edges.
- Deleting unnecessary edges has a computational cost associated with it, but this is likely minimal compared to the cost associated with exact graph comparison.
- If ever the network was to be re-structured, all of the information is already present.
- If AGs remain small (less than 100 nodes), then the computational time for graph comparison may not be crippling, with sufficient resources.

Points to consider for the community comparison approach:

- Edges are again dependent on the structures chosen.
- A lot of the edges between communities will be redundant, but again, the number of such edges will on average be less than in the fully-connected network.
- This approach still does not guarantee that structures end up in the correct community.

The community comparison approach seems like a promising step towards reducing the computational cost associated with generating the network. However, there are clear questions about how likely it is to misclassify structures.

One improvement on this approach could be to include some prior belief of which community a new structure is likely to belong to before moving to a full comparison. The prior belief could be formed by comparing the number of nodes and attributes for graphs, since these take a relatively short time to compare. Once these initial guesses have been computed, an exact graph matching algorithm could then be applied to give a definitive classification.

2.6 Database approaches

'Neurons that fire together, wire together' - Carla Shatz [13] (paraphrasing Donald Hebb [14]—'When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased (p. 62).)

Using this it is possible to develop an alternative idea of how to determine the strength of connections in the network. Since the network will one day underlie a database of structures, it will be subject to numerous queries. If structures are often featured in queries together, surely they must share some similarity? Using this approach, after some time, the strength of connections will become something that users can query to discover similar structures.

It would be entirely possible to leave the database unconnected until the first user comes in with the first search; however, a more efficient approach may be to query the database when new structures are added (the queries could

be put on hold until a time when there is little traffic). When a structure is added, it could query the database to find other structures that share the same attributes in order to find the structures it is similar too.

These attributes represent inherent properties of the structures, and have the potential to create a complex network.

One could implement some form of pruning and forgetting mechanism to ensure that weights do not become excessive. Additionally, structures may appear very similar when few examples of their particular class exist within the database, as there are no others which are a closer match. For example a slab bridge and a suspension bridge, added to a database with few other bridges, may appear close. However, when another slab and beam bridge is added, the strength of the connection between the slab bridge and the suspension bridge should weaken as the two beam and slab bridges appear more similar.

Aside from attributes, metadata describing the graph invariants for graphs could become a useful target for searches. This is the basis for spectral graph theory and these invariants are cheaper to compute than running an exact graph matching algorithm.

Graph comparisons (e.g. which graphs contain a specific subgraph) would still feature in this, although of course the computational cost of such operations is to be avoided where possible. The user could specify full graph comparisons on structures they believe to be suitable candidates. The weight of the connection between these structures could then be adjusted according to the strength of the match. A weak match would reduce the strength of the connection, while a strong match would increase the strength of the connection.

3 COMPUTATIONAL COMPLEXITY FOR VARIOUS GRAPH COMPARISON METHODS

Some of the methods of comparing attributed graphs are more computationally expensive than others; however, often a faster computational speed often comes at the cost of reduced information. The fastest comparison one may run is asking how similar are the two graphs in terms of size, i.e. do they have a similar number of nodes and edges? This is a trivial operation to run, and for identical graphs, the number would naturally be the same. However, it is clear that asking if two graphs are the same size cannot provide any information about the topological similarity or whether or not the graphs have similar attributes.

At the opposite end of the scale in terms of computational cost would be an exact graph comparison, finding the maximum common subgraph. For a review on methods and their relative computational costs, the reader is directed towards [15]. All methods are limited by the fact that finding the maximum common subgraph (also known as the subgraph isomorphism problem) is NP-complete. The benefit of these approaches, however, is that they can provide a list of all possible subcomponents shared by two structures. Using these methods, it is also possible to match not just the topology of the two graphs, but also the attributes.

The spectra of the Laplacian matrix for two different graphs can be used as a measure of their topological similarity [8]. The measure of similarity is obtained by calculating distance between the Laplacian spectra of two graphs. This method is more sensitive to topological similarities, but still returns a single number.

Neural network approaches provide vector embeddings of the graph, allowing the distance to be calculated as the difference between two vectors. A neural network approach has been used before for examining the similarity of different functions by representing the functions as control flow graphs [11]. One could say similar things about GNN-based approaches, although some of the attention-based methods do highlight where similarities lie.

Of course these different methods can be combined to create a distance matrix that incorporates various measures of similarity between the graphs. This raises the question of how best to combine the various distance metrics. Another possible avenue to explore is if one can predict the MCS between two graphs using the information from other methods?

Using this approach would still require a pair-wise comparison using the simpler distance metrics, but could reduce computational time by highlighting candidate pairs that were likely to share a large MCS.

4 CONCLUSIONS

Hopefully this paper has provided an insight into the challenge of initiating connections in a network where existing connections do not in fact exist. Future work will focus on developing the approaches described in Section 2 and, once the IE models have been included in the database, testing these approaches to find which ones work best.

The merits of various methods for graph comparison have also been described, and again, it remains to be seen which ones provide sufficient information for the comparison to be useful in transfer learning. Future work will focus on testing these methods, but also on developing new measures and methods of comparison that are more suited to the problem at hand.

ACKNOWLEDGEMENTS

The authors would like to thank the UK EPSRC for funding through the Established Career Fellowship EP/R003645/1 and the Programme Grant EP/R006768/1.

REFERENCES

[1] L. Bull, P. Gardner, J. Gosliga, T. Rogers, N. Dervilis, E. Cross, E. Papatheou, A. Maguire, C. Campos, and K. Worden, "Foundations of population-based SHM, part I: homogeneous populations and forms," *Mechanical Systems and Signal Processing*, 2020.

[2] J. Gosliga, P. Gardner, L. Bull, N. Dervilis, and K. Worden, "Foundations of population-based SHM, part II: heterogeneous populations – graphs, networks, and communities," *Mechanical Systems and Signal Processing*, 2020.

[3] P. Gardner, L. Bull, J. Gosliga, N. Dervilis, and K. Worden, "Foundations of population-based SHM, part III: heterogeneous populations – transfer and mapping," *Mechanical Systems and Signal Processing*, 2020.

[4] I. Antoniadou, N. Dervilis, E. Papatheou, A. Maguire, and K. Worden, "Aspects of structural health and condition monitoring of offshore wind turbines," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 373, 2015.

[5] E. Papatheou, N. Dervilis, A. Maguire, I. Antoniadou, and K. Worden, "A performance monitoring approach for the novel Lillgrund offshore wind farm," *IEEE Transactions on Industrial Electronics*, vol. 62, pp. 6636–6644, 2015.

[6] M. El-Mehalawi and R. A. Miller, "A database system of mechanical components based on geometric and topological similarity. part I: representation," *Computer Aided Design*, vol. 35, pp. 83–94, 2001.

[7] M. El-Mehalawi and R. A. Miller, "A database system of mechanical compoennts based on geometric and topological similarity. part II: indexing, retrieval, matching, and similarity assessment," *Computer Aided Design*, vol. 35, 2001.

[8] Y. Shimada, Y. Hirata, T. Ikeguchi, and K. Aihara, "Graph distance for complex networks," *Scientific Reports*, 2016.

[9] I. Koch, "Enumerating all connected maximal common subgraphs in two graphs," *Theoretical Computer Science*, vol. 250, pp. 1–30, 2001.

[10] Y. Cao, T. Jiang, and T. Girke, "A maximum common substructure-based algorithm for searching and predicting drug-like compounds," *Intelligent Systems for Molecular Biology*, vol. 24, 2008.

[11] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *Proceedings of the 36^{th} International Conference on Machine Learning*, 2019.

[12] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, pp. 575–577, 1973.

[13] C. J. . Shatz, "The developing brain," *Scientific American*, vol. 267, 1992.

[14] D. Hebb, *The Organisation of Behaviour*. John Wiley and Sons, 1949.

[15] E. Duesbury, J. Holliday, and P. Willett, "Maximum common subgraph isomorphism algorithms," *MATCH Communications in Mathematical and in Computer Chemistry*, vol. 77, pp. 213–232, 2017.