Proceedings Paper:
Gosliga, J., Gardner, P.A., Bull, L.A. et al. (2020) Towards population-based structural health monitoring, Part III: graphs, networks and communities. In: Dilworth, B.J. and Mains, M., (eds.) Topics in Modal Analysis & Testing. Proceedings of the 38th IMAC, A Conference and Exposition on Structural Dynamics 2020. 38th International Modal Analysis Conference, 10-13 Feb 2020, Houston, TX, USA. Conference Proceedings of the Society for Experimental Mechanics Series, 8. Springer, pp. 255-267. ISBN: 9783030477165. ISSN: 2191-5644.

https://doi.org/10.1007/978-3-030-47717-2_26

Towards population-based structural health monitoring, Part III: Graphs, networks and communities

J. Gosliga, P.A. Gardner, L.A. Bull, N. Dervilis and K. Worden

Dynamics Research Group, Department of Mechanical Engineering
University of Sheffield, Sheffield S1 3JD, UK

Abstract

Population-based structural health monitoring opens up the possibility of using information from a population of structures to provide extra information for each individual structure. For example, population-based structural health monitoring could provide improved damage-detection within a homogeneous population of structures by defining a normal condition across a population of structures, which was robust to environmental variation. Furthermore, in cases where structures are sufficiently similar, damage location, assessment, and classification labels could be transferred, increasing the damage labels available for each structure. To determine whether two structures are sufficiently similar requires the comparison of some representation of the structure. In fields such as bioinformatics and computer science, attributed graphs are often used to determine structural similarity. This paper will describe methods for comparing the topology attributes of two such graphs. The algorithm described is suited to population-based structural health monitoring as it provides matches between two graphs which have physical significance. This paper will also describe the process of comparing hierarchical attributes to determine the level of knowledge transfer possible between two structures.

Keywords: Population-based structural health monitoring; Irreducible Element model; Attributed Graph; structural similarity

1 Introduction

The ideal case for structural health monitoring (SHM) is where both the normal condition and the various damage states for a structure are known, and corresponding data sets are available. If the normal condition (including any benign variations) is known, then damage detection is trivial, since any deviation must come from damage to the structure. Additionally, if data were available for all possible damage states, then for any signal measured from the structure the damage location and classification labels would be known. Also, if detailed data for each damage state were available, it would be possible to determine the extent of the damage. A situation where damage can be detected, classified, located, and the extent calculated would be extremely favourable, and with the aid of a good model could allow for damage prediction [1].

In reality, data sets are scarce, for both the normal condition and the damage states of an individual structure. A solution to this is to move away from individuals and perform structural health monitoring using populations of structures instead [2, 3]. Population-based structural health monitoring (PBSHM) seeks to overcome the limitation on available data for structural health monitoring, which to date has been a major obstacle in the industrial uptake of SHM. By examining a population of structures, rather than simply looking at each structure individually, it may be possible to share information about the normal condition and damage states, provided certain conditions are met. An example of this is the Lillgrund wind farm [4], where the nominally identical structures form a homogeneous population [5]. In the Lillgrund wind farm, it was possible to make the detection of performance anomalies robust to variations in the normal condition by sharing SCADA data between wind turbines. In homogeneous populations, it is

also possible to describe the population using a single model, called a *form*, which effectively captures the variation within the population [5].

Homogeneous populations are a special case within PBSHM, as for a population to be homogeneous, all structures within the population must have identical topology, and nominally-identical geometry and materials. For a population to be considered strongly homogeneous, the boundary conditions also need to match. Otherwise, a population is considered to be heterogeneous, which has implications for the level of information transfer possible [6]. Within a heterogeneous population, certain members may share a greater or lesser degree of similarity with another. The degree of similarity between two structures will determine the information transfer possible. In order to calculate the degree of similarity between structures, it is first necessary to create an Irreducible Element (IE) model of the structure (as described in [7]) and then convert this IE model into an attributed graph (AG) [8]. This paper will describe how the AGs from two separate structures are compared in order to determine the degree of similarity.

This paper will also describe the *network* of structures, which is formed from the AGs of various structures. The idea of the network is that it will create a space where the distance between members of a heterogeneous population is determined by the degree of similarity [9]. Since the distance is determined by the degree of similarity between two structures, it is expected that similar structures will be closer together, and that this can be used to create *communities* of structures. Within communities, the structures should be similar enough that information transfer is possible. For example, within a community of aeroplanes, some information regarding damage in a wing should be transferable, as all aeroplanes have wings.

Since the distance between structures in the network is determined by how similar they are, it is necessary to have some method for quantifying structural similarity. The first step in quantifying the structural similarity is describing the objects as IE models. These IE models are then converted into an AG. It is then possible to use AGs for two structures to find common substructures or *common subgraphs*. For this application, the search is limited to *induced* and *connected* subgraphs. Thankfully the problem of finding common subgraphs to quantify structural similarity exists in other fields – most notably chemistry, bioinformatics and computer science – which means that tools are already available [10]. Methods for finding the induced common subgraph [11] and connected induced common subgraph [12] will be explored in this paper. A method for calculating a similarity score based on the size of the subgraph and attribute matching, relevant to this application, will be developed.

2 Matching using the attributed graphs

An IE model of a structure is a way of abstracting and representing aspects of a structure that are important for transferring information about that structure. By comparing the IE representations of two structures, it is possible to determine the level of inference possible between them. To allow this comparison to be performed automatically, it is more efficient to restructure the information into the form of an AG. In the AG, the same information is used as for the IE representation, but whereas the properties for the IE representation are organised in a tabular format to improve readability by humans, the information in the attributed graph is organised so that it can be more efficiently processed by a graph-matching algorithm. The process for creating the AG for a structure is illustrated in Fig. 1 and described in [7, 8].

For this paper, it is assumed that the previous steps have been followed, generating the attributed graphs for an aeroplane and a wind turbine as shown in Fig. 2. The attributes are taken from the IE representations; where the elements and joints for the aeroplane come from Tables 2 and 4 respectively, and the elements and joints for the wind turbine come from Tables 1 and 3.

2.1 Introduction to graph-matching algorithms

Definition

A graph G is defined by a set of nodes V and edges E, and so $G = (V, E)$ [13]. The edge and node set of a graph can alternatively be represented as $E(G)$ and $V(G)$, respectively. It is also possible to define a graph by its neighbourhood N, where the neighbourhood is defined as $N(v) = \{u \in V \mid (u, v) \in E\}$. This is a common approach when writing code which incorporates graphs, and an example of a graph defined using the dictionary datatype in Python is shown

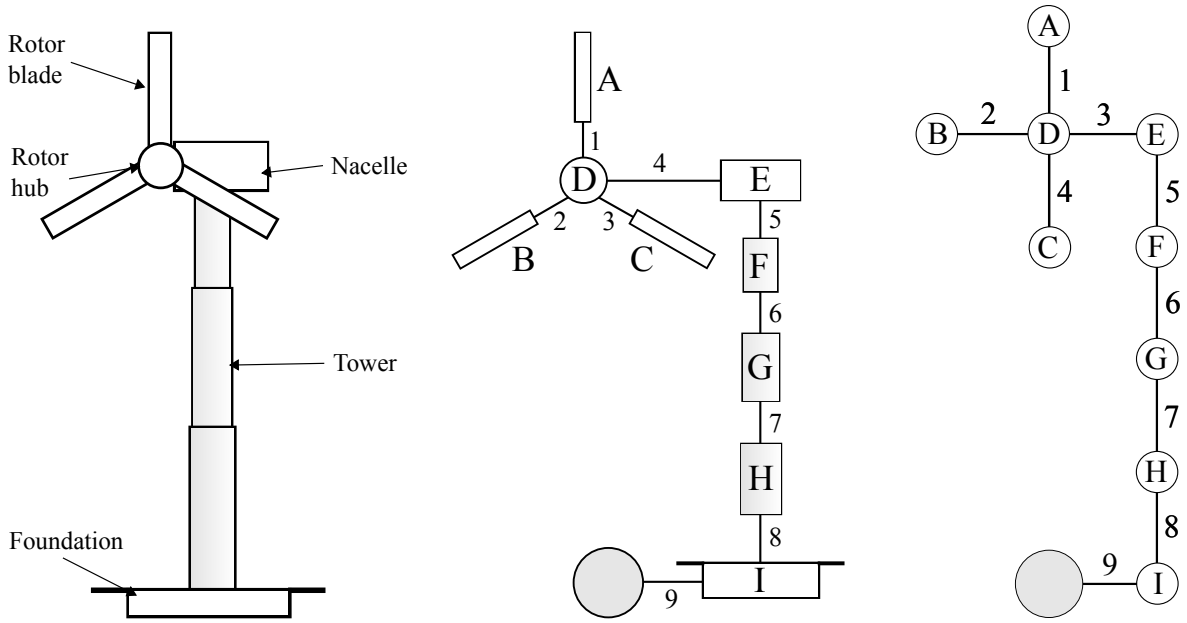


Figure 1: The evolution of an attributed graph. The structure in this case is a wind turbine, and shown (from left to right) are the IE model; an expanded version of the IE model, showing the element and joint designations; and finally, the AG.

in Fig. 3. The neighbourhood N as defined does not contain the node itself; this is known as the *open neighbourhood* of v. The *closed neighbourhood* of a node v, contains the node itself and is defined as $N[u] = N(v) \cup v$. Neighbourhoods are assumed to be open unless otherwise specified. If two nodes from $V(G)$, say v_1 and v_2, are adjacent, then $(v_1, v_2) \in E(G)$. A graph is *connected* if there is a path from any node to any other node.

Isomorphism

Two graphs, G and H, are said to be isomorphic if there exists a one-to-one mapping from $V(G) \to V(H)$, and a one-to-one mapping from $E(G) \to E(H)$. Isomorphic graphs have the same topology, but have different node labels. Therefore, all structures within a homogeneous population will have isomorphic AGs.

Subgraphs

A graph G' is said to be a subgraph of G, if $G' \subseteq G$, which implies $V' \subseteq V$ and $E' \subseteq E$. There are different types of subgraph, which are differentiated by the rules for generating the subset of edges E', given the subset of nodes V'. For example, given an arbitrary subset of nodes V', an *induced* subgraph of G is a subgraph where the edge set E' contains all the edges from E that have both endpoints in V', formally $E' = \{(u, v) \in E \mid u, v \in V'\}$. Induced subgraphs are the most useful for this particular application. Two graphs, G and H, share a *common* subgraph if both $G' \subseteq G$ *and* $G' \subseteq H$. Finding the largest common induced subgraph is a problem that has already been solved [11], as is finding the maximum common connected induced subgraph [12]. For this application, only connected induced subgraphs will be considered. If an AG falls into two disconnected components, it means it is representing two disconnected structures, and the two structures should be considered separately.

The problem of finding the maximum common induced subgraph between two graphs G and H is NP-complete, meaning that exhaustive search algorithms are required to solve it. The maximum common induced subgraph problem is a generalisation of the subgraph isomorphism problem. The subgraph isomorphism problem involves taking two graphs G and H and determining whether or not G contains a subgraph that is isomorphic to H. The subgraph isomorphism problem can be solved by finding the maximal cliques in the modular product of the two graphs, where a clique is a subset of nodes of an undirected graph such that every two distinct nodes in the clique are adjacent. Since the problem of enumerating all maximal cliques is NP-hard, the computational time of any algorithm used to

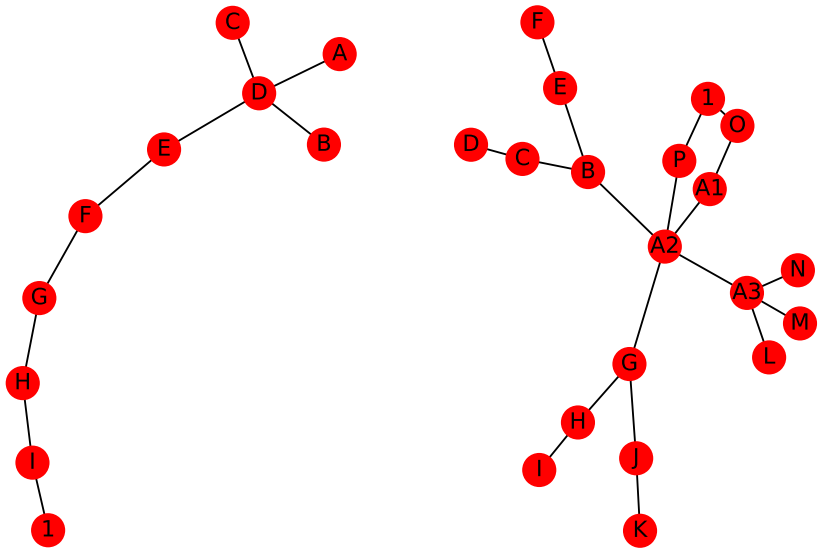
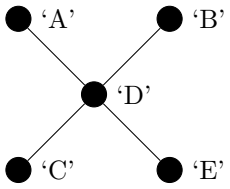


Figure 2: The graphs prodcued from list of elements in Tables 1 and 2 and list of joints in Tables 3 and 4. The graph shown on the left is from the wind turbine (compare with Fig. 1) and the graph on the right is from the aeroplane.



graph = { 'A' : ['D']
'B' : ['D']
'C' : ['D']
'D' : ['A', 'B', 'C', 'E']
'E' : ['D'] }

Figure 3: A graph and the corresponding Python dictionary. The dictionary defines the neighbourhood for each node.

solve this problem will not scale for larger graphs. However, the graphs for structures are unlikely to become so large that this is an issue.

The modular product graph

The *modular product* is a graph (represented by $G \diamond H$) produced by combining two graphs G and H. An example of a modular product graph is shown in Fig. 4. The modular product has the node set, generated by the cartesian product $V(G) \times V(H)$ in which two nodes (x, u) and (y, v) are adjacent if:

- $(x, y) \in E(G)$ and $(u, v) \in E(H)$, or
- $(x, y) \notin E(G)$ and $(u, v) \notin E(H)$

Examining Fig. 4 it can be seen that the cliques correspond to subgraphs which are common to both G and H. For example, the nodes A_1B_2 and A_2B_3 in the modular product correspond to the subgraphs G' and H', where $V(G') = \{A_1, A_2\} \subseteq V(G)$ and $V(H') = \{B_1, B_2\} \subseteq V(H)$, with corresponding edges $(A_1, A_2) \in E(G)$ and $(B_1, B_2) \in E(H)$. It is possible to verify that this is an induced subgraph.

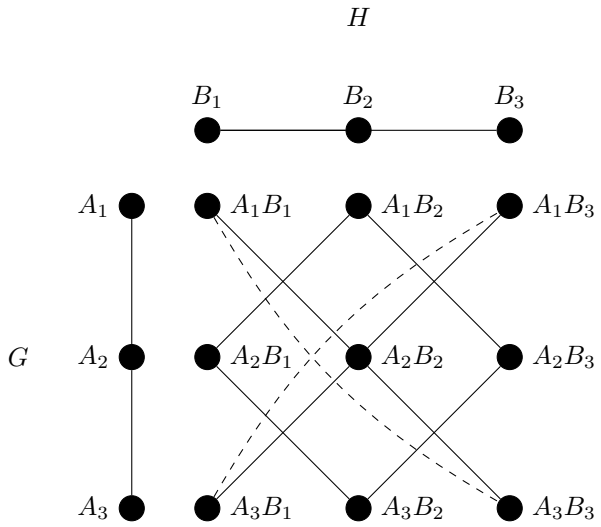


Figure 4: The modular product graph for G and H. The solid lines represent c-edges and the dashed lines represent d-edges.

However, induced subgraphs are not always connected. If the nodes A_1, B_1 and A_3, B_3 were chosen instead, then $V(G') = \{A_1, A_3\}$. There are no edges in $E(G)$ with both endpoints in $V'(G)$, since the edge (A_1, A_3) is not in $E(G)$. The subgraph G' is still induced, but since no edges exist in G', it is unconnected. Likewise for the subgraph H', where $V(H') = \{B_1, B_3\}$.

2.2 Bron-Kerbosch algorithm

As mentioned previously, finding cliques in the modular product of two graphs is equivalent to finding the common subgraphs. The Bron-Kerbosch algorithm [11], described in Algorithm 1, is a clique-finding algorithm that will report all maximal cliques, and therefore all subgraphs between two graphs. This algorithm is guaranteed to generate induced subgraphs but does not guarantee that they are connected. The Bron-Kerbosch algorithm is considered one the most efficient backtracking algorithms for maximal clique enumeration is widely used in chemical and life science applications [10]. The complexity of the Bron-Kerbosch algorithm is $\mathcal{O}(3^{n/3})$.

Algorithm 1 Bron-Kerbosch

R: set of nodes to be reported, initially $R = \emptyset$
P: set of nodes which can be added to R, initially $P = V$
X: set of nodes which cannot be added to R, initially $X = \emptyset$

```
1: procedure EnumerateCliques(R, P, X)
2:     Let P be the set {u_1, ..., u_k}
3:     if P = ∅ and X = ∅ then
4:         report R
5:     else
6:         for i ← 1 to k do
7:             P ← P\{u_i}
8:             N ← {v ∈ V | (u_i, v) ∈ E}
9:             EnumerateCliques(R ∪ {u_i}, P ∩ N, X ∩ N)
10:            X ← X ∪ {u_i}
11:        end for
12:    end if
13: end procedure
```

The algorithm will report all of the cliques in the graph. In the graph shown in Fig. 5, these will be $\{A, B, C\}$ and

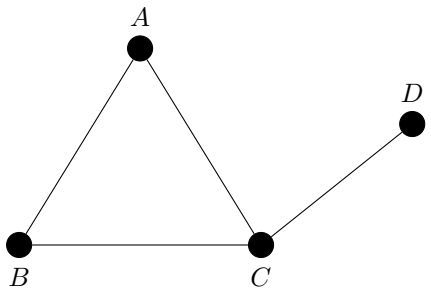


Figure 5: A simple graph for illustrating clique-finding algorithms.

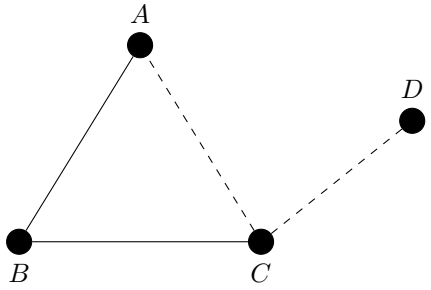


Figure 6: A simple graph for illustrating clique-finding algorithms. The solid lines represent c-edges and the dashed lines represent d-edges.

$\{C, D\}$. To find the largest subgraph, the cliques are then ranked in order of size. Due to node re-labelling any clique-finding algorithm will report all graphs that are isomorphic to a given subgraph. The alignment of the subgraph which gives the best physical match is made possible through the use of attributes.

2.3 C-clique-finding algorithm

This algorithm is a modification of the Bron-Kerbosch algorithm which ensures only connected induced subgraphs are found. This is not only useful for ensuring they are valid in a physical sense, but also significantly cuts down the search time. In order to limit the search to connected subgraphs, it is first necessary to define the concept of *c-edges*.

C-edges

Two nodes (x, u) and (y, v) in the modular product $G \diamond H$ are adjacent via a c-edge (where the c stands for connected) if $(x, y) \in E(G)$ and $(u, v) \in E(H)$. This is the first condition for adjacency in the modular product graph. The rest of the edges are d-edges (where the d stands for disconnected) and are defined by the second adjacency condition in the modular product graph. The c-edges and d-edges are shown in Fig. 4.

Cliques which include c-edges are called c-cliques. Limiting the clique-finding algorithm to only find c-cliques ensures that any reported subgraphs are connected. This can also decrease the computation time [12] as the set of c-cliques is a subset of the set of all cliques in a graph. Therefore a smaller number of cliques needs to be considered, which reduces the size of the search tree. If the graph contains d-edges, as shown in Fig. 6, the only clique of interest is $\{A, B, C\}$ as the clique $\{C, D\}$ does not contain any c-edges.

To limit the search to only c-cliques, it is necessary to modify Algorithm 1. The major modification is the creation of a new set D which contains the set of nodes which cannot be directly added to the current clique because they are adjacent to the current node u via a d-edge.

This problem now requires an initialisation algorithm, shown in Algorithm 2. This algorithm iterates over all nodes in the graph, gradually adding them to T, which is the set of nodes that cannot be added to a clique because they have

already been used to intialise the algorithm. It is necessary to exclude nodes that have already been used to initialise the algorithm because otherwise every clique of size n would be reported n times.

Algorithm 2 Initialisation for Enumerate C-Cliques

R: set of nodes to be reported
P: set of nodes which can be added to R, because they are adjacent to node u via c-edges
D: set of nodes which cannot be directly added to R, because they are adjacent to u via d-edges
T: set of nodes which have already been used to initiate the EnumerateC-Cliques algorithm
E_c: set of c-edges for the graph G E_d: set of d-edges for the graph G

```
1: for u ∈ V do
2:     P ← ∅
3:     D ← ∅
4:     X ← ∅
5:     N ← {v ∈ V | (u, v) ∈ E}
6:     for v ∈ N do
7:         if (u, v) ∈ E_c then
8:             if v ∈ T then
9:                 X ← X ∪ {v}
10:            else
11:                P ← P ∪ {v}
12:            end if
13:        else if (u, v) ∈ E_d then
14:            D ← D ∪ {v}
15:        end if
16:    end for
17:    R ← {u}
18:    EnumerateC-Cliques(R, P, D, X, T)
19:    T ← T ∪ {u}
20: end for
```

The initialisation algorithm iterates through all nodes in the neighbourhood $N(v)$ and determines whether a node u is adjacent to v via a c-edge, in which case – provided that node has not been previously used to initialise EnumerateC-Cliques – the node is added to P. Nodes in P may be directly added to R in the next procedure call. If the node u has been previously used to initialise EnumerateC-Cliques, it is added to X instead. If the node u is adjacent to v via a d-edge, it is added to D. The node u is then added to R and the procedure EnumerateC-Cliques, described in Algorithm 3 is called. After the procedure has finished, u is added to T so that it is excluded from being added to future cliques. This procedure excludes cliques which do not contain a simple path of c-edges that connect all vertices within that clique (for example, $\{C, D\}$ in Fig. 6).

A node in D cannot be added to R at the point where EnumerateC-Cliques is called. However, nodes in D may at some point be adjacent to a node in P via a c-edge. This check is performed on Lines 9 and 10 in Algorithm 3, where each node in D is checked for adjacency via a c-edge with the current node u_i. If the node v is adjacent to u_i via a c-edge, and has not previously been used to intialise the procedure EnumerateC-Cliques (i.e. $v \notin T$), then the node is added to P and can potentially be added to R in the next procedure call. If the node $v \in T$ then v is added to X. The node is then removed from D. Aside from this this check to see whether or not nodes can be moved from D to P, Algorithm 3 is essentially the same as Algorithm 1.

For the structures shown in Fig. 2, the maximum common connected induced subgraph found by Algorithm 3 is shown in Fig. 7. For the remainder of the discussion all subgraphs are assumed to be connected and induced.

3 Similarity scores

The similarity scores are calculated by examining the node attributes in each graph that form the subgraph in each question. This is achieved by extracting the nodes in the subgraph and querying the dictionary containing the node

Algorithm 3 Modified Bron-Kerbosch

R: set of nodes to be reported
P: set of nodes which can be added to R, because they are adjacent to node u via c-edges
D: set of nodes which cannot be directly added to R, because they are adjacent to u via d-edges
T: set of nodes which have already been used to initiate the EnumerateC-Cliques algorithm
E_c: set of c-edges for the graph G

```
1: procedure EnumerateC-Cliques(R, P, D, X, T)
2:     Let P be the set {u_1, ..., u_k}
3:     if P = ∅ and X = ∅ then
4:         report R
5:     else
6:         for i ← 1 to k do
7:             P ← P\{u_i}
8:             N ← {v ∈ V | (u_i, v) ∈ E}
9:             for all v ∈ D do
10:                if (v, u_i) ∈ E_c then
11:                    if v ∈ T then
12:                        X ← X ∪ {v}
13:                    else
14:                        P ← P ∪ {v}
15:                    end if
16:                    D ← D\{v}
17:                end if
18:            end for
19:            EnumerateC-Cliques(R ∪ {u_i}, P ∩ N, D ∩ N, X ∩ N, T)
20:            X ← X ∪ {u_i}
21:        end for
22:    end if
23: end procedure
```

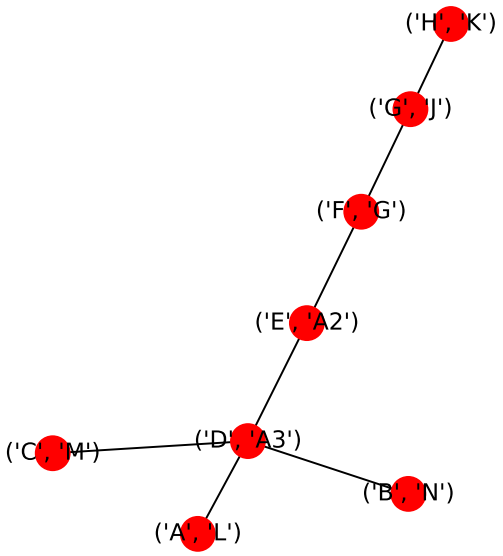


Figure 7: The maximum common connected induced subgraph of the two graphs shown in Fig. 2.

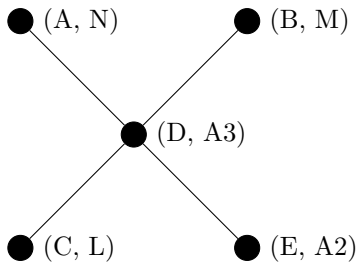


Figure 8: A possible subgraph from the two graphs shown in Fig. 2.

attributes. For the sample subgraph shown in Fig. 8, the node labels can be used to look up the attributes for the corresponding nodes in the original AGs, shown in Fig. 2.

The left-hand entries in node labels (A, B, C, D, E) correspond to the wind turbine, and so node and edge attributes can be found in Tables 1 and 3, whereas the right-hand entries in the node labels (A2, A3, N, M, L) correspond to the aeroplane and can be used to find node attributes from Tables 2 and 4. For example, the geometry class and shape for node in the wind turbine is 'Beam, Aerofoil', and for node N in the aeroplane the geometry class and shape is also 'Beam, Aerofoil' so there is a match. However, node D in the wind turbine represents *complex* geometry, while node A3 in the aeroplane represents a *shell*, so these do not match.

The comparison is performed at the greatest possible level of resolution, if the geometry class matches, then the shape is compared. If the shape matches, then the dimensions for each are examined. If the elements match on all levels, then they are essentially identical. This is shown as a flowchart in Fig. 9. This is true for the case of materials as well. If the material class, specific material and material properties are all the same, then the two elements are made from the same material.

The same matching can be performed for joints. In this case, the joint to examine in each graph is determined by examining the edges in the subgraph. which are adjacent in the subgraph. This gives a set of nodes which are connected by the edge in the original graphs. From this, the edge attributes can be extracted. The joint type is compared, and if the joint is kinematic, the restricted degrees of freedom can also be compared.

If all elements and joints in two graphs are identical (all of the hierarchical attributes are available and match) then these two graphs form a homogeneous population. If all elements and joint in the subgraph of two graphs are identical, then they share a subcomponent and transfer is possible within this subcomponent [6]. The more likely case is that only partial matches are seen, and in this case the level of match determines the appropriate transfer learning approach.

4 Communities

It is possible to determine a similarity score (Jaccard similarity coefficient) based purely on the size of this maximum common subgraph G'. This is calculated using the following equation,

$$J_v(G,H) = \frac{\mid V(G') \mid}{\mid V(G) \mid + \mid V(H) \mid - \mid V(G') \mid} \tag{1}$$

where $J_v(G,H)$ is the Jaccard index for the node sets. Similarly, it is possible to find the Jaccard index for the edge sets,

$$J_e(G,H) = \frac{\mid E(G') \mid}{\mid E(G) \mid + \mid E(H) \mid - \mid E(G') \mid} \tag{2}$$

Multiplying $J_v(G,H)$ and $J_e(G,H)$ by 100 gives a percentage similarity score. This was calculated for several representative structures and the results are shown in Fig. 10. Structures tend to match more with similar structures, for example aeroplanes match strongly with aeroplanes, and bridges match strongly with bridges. However, some

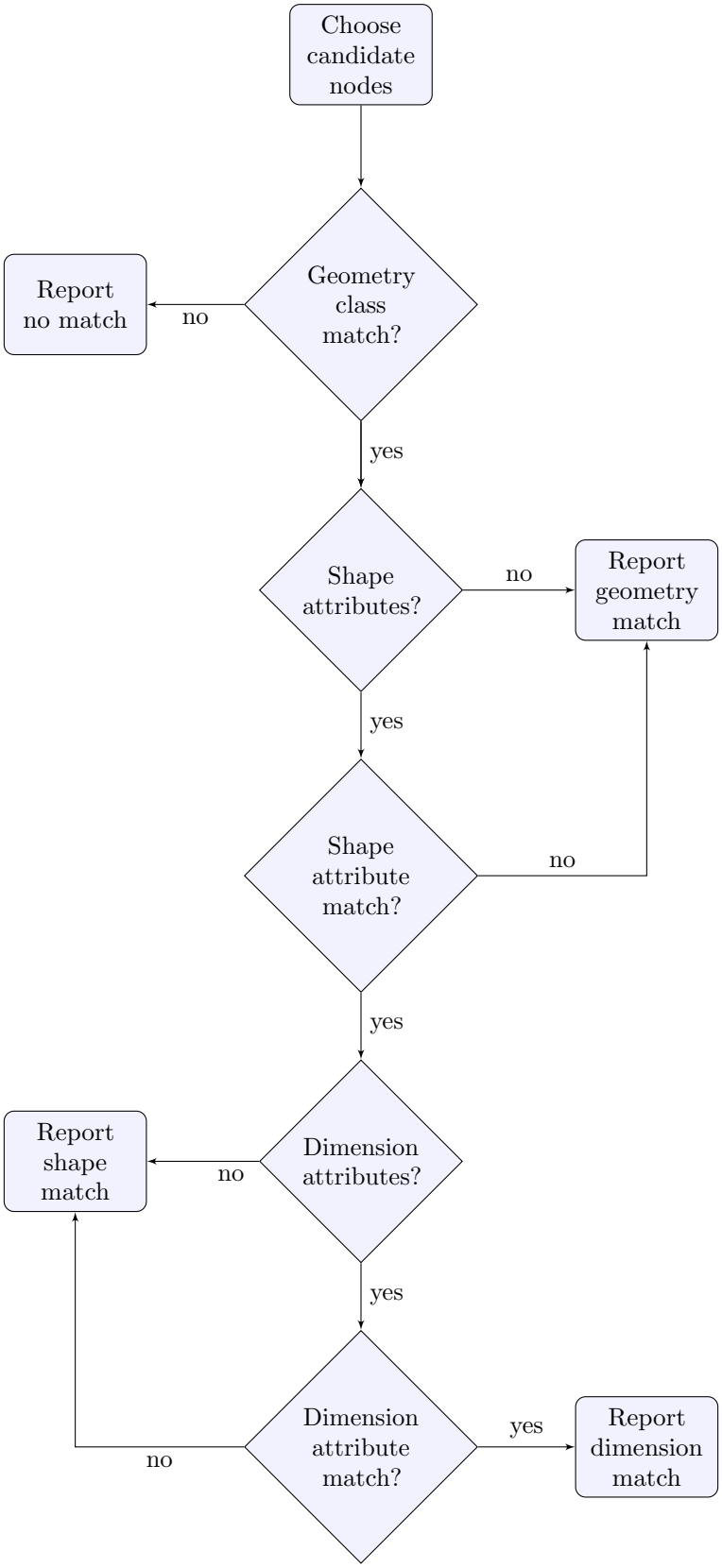


Figure 9: A flowchart for comparison of the geometry attributes of a pair of nodes in the AG.

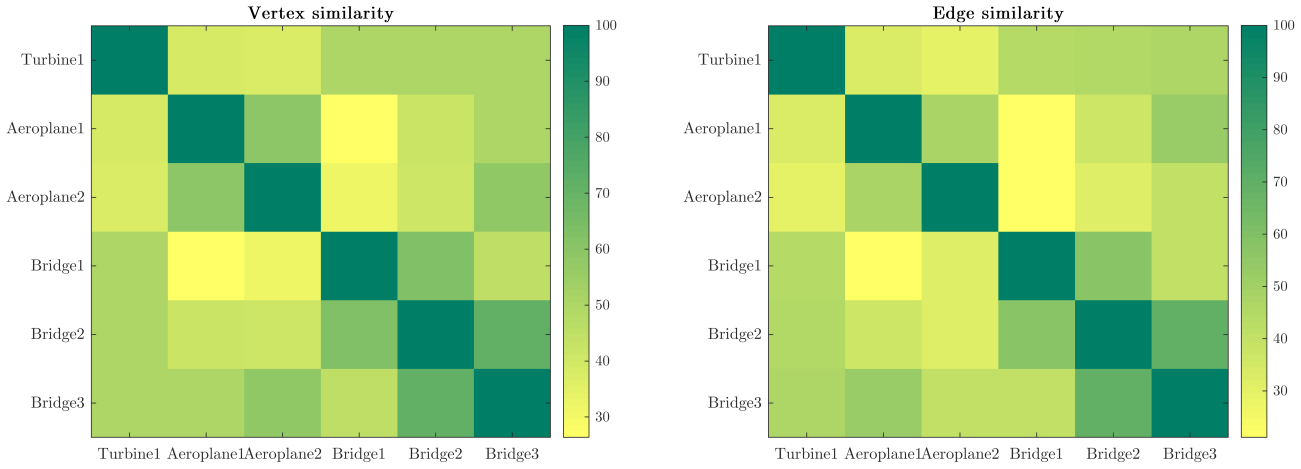


Figure 10: Figures showing the percentage topology match between structures. The percentage score is based on the Jaccard index $J_v(G, H)$ and $J_e(G, H)$ for nodes and edges respectively.

bridge graphs match more with the aeroplane and turbine structures. This could be caused by there being a higher chance of finding large subgraphs when the two graphs G and H are larger.

This highlights one of the main issues with matching on topology alone. Topology aids in finding similar structures with consistent location labels, but topology matching alone does not guarantee that nodes in the subgraph have similar geometry or material attributes. For example, looking at topology alone may lead one to match the deck of a bridge with the wing of an aeroplane. Equally, when matching an aeroplane with an aeroplane there are many orientations of the subgraph that do not produce a sensible match. For example, the fuselage may be aligned with a wing of the aeroplane. The key to producing sensible matches is using similarity scores which take the node attributes into account.

By grouping structures according to their similarity it should be possible to create *communities* of structures. Within these communities, it is expected that a certain level of information transfer is possible. The communities which are formed may change based on the particular SHM problem, depending on which attributes are determined to be the most relevant.

5 Conclusion

To compare the similarity of two structures, they are first converted into IE models. The IE models are then converted into AGs. These graphs are then compared to find a list of possible substructures. The method for doing this is to first find the modular product of the two graphs, and then use a clique finding algorithm to generate a list of subgraphs. The attributes of the corresponding nodes in each graph can then be compared. Comparing node attributes is simple as the nodes in each graph are uniquely identified (within that graph) and the information can be stored using a Python dictionary, or in a database. Transferring this methodology to a database will form the basis of future work.

6 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] A. Rytter. *Vibrational Based Inspection of Civil Engineering Structures*. PhD thesis, Deptartment of Building Technology and Structural Engineering, Aalborg University, Denmark, 1993.

[2] I. Antoniadou, N. Dervilis, E. Papatheou, A.E. 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*, 373, 2015.

[3] K. Worden, E.J. Cross, N. Dervilis, E. Papatheou, and I. Antoniadou. Structural health monitoring: from structures to systems-of-systems. *IFAC-PapersOnLine*, 48:1–17, 2015.

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

[5] L.A. Bull, P.A. Gardner, J. Gosliga, T.J. Rogers, M. Haywood-Alexander, N. Dervilis, E.J. Cross, and K. Worden. Towards population-based structural health monitoring, Part I: Homogeneous populations and forms. In *Proceedings of IMAC XXXVIII – the 38th International Modal Analysis Conference, Houston, TX*, 2020.

[6] P.A. Gardner and K. Worden. Towards population-based structural health monitoring, Part IV: Heterogeneous populations, matching and transfer. In *Proceedings of IMAC XXXVIII – the 38th International Modal Analysis Conference, Houston, TX*, 2020.

[7] J. Gosliga and K. Worden. A general representation for assessing the similarity of structures. In *Proceedings of the 12th International Workshop on Structural Health Monitoring, Palo Alto, CA*, 2019.

[8] J. Gosliga, P.A. Gardner, L.A. Bull, N. Dervilis, and K. Worden. Towards population-based structural health monitoring, Part II: Heterogeneous populations and structures as graphs. In *Proceedings of IMAC XXXVIII – the 38th International Modal Analysis Conference, Houston, TX*, 2020.

[9] K. Worden. Towards population-based structural health monitoring, Part VI: Structures as geometry. In *Proceedings of IMAC XXXVIII – the 38th International Modal Analysis Conference, Houston, TX*, 2020.

[10] E. Duesbury, J.D. Holliday, and P. Willett. Maximum common subgraph isomorphism algorithms. *Match*, 77:213–232, 2017.

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

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

[13] R. Diestel. *Graph Theory, 3rd ed.* Springer-Verlag, New York, 2006.

A Appendix

Tables listing the element and joint properties for the IE models.

Table 1: List of elements and their properties for Turbine 1

Element designations for Turbine 1				
Name	**Element ID**	**Material**	**Geometry**	**Shape**
Rotor blade	A	FRP	Beam	Aerofoil
Rotor blade	B	FRP	Beam	Aerofoil
Rotor blade	C	FRP	Beam	Aerofoil
Rotor hub	D	FRP	Complex	Rotor hub
Nacelle	E	FRP	Shell	Cuboid
Tower section 1	F	Metal	Beam	Cylindrical
Tower section 2	G	Metal	Beam	Cylindrical
Tower section 3	H	Metal	Beam	Cylindrical
Foundation	I	Concrete	Plate	Cylindrical
Name	**Element ID**	**Boundary**	-	-
Footing	1	Ground	-	-

Table 2: List of elements and their properties for Aeroplane 1

Element designations for Aeroplane 1				
Name	**Element ID**	**Material**	**Geometry**	**Shape**
Fuselage	A1	FRP	Shell	Truncated cone
Fuselage	A2	FRP	Beam	Cylindrical
Fuselage	A3	FRP	Shell	Cone
Wing 1	B	FRP	Beam	Aerofoil
Pylon 1	C	FRP	Complex	Pylon
Engine 1	D	Assembly	Shell	Cylinder
Pylon 2	E	FRP	Complex	Pylon
Engine 2	F	Assembly	Shell	Cylinder
Wing 2	G	FRP	Beam	Aerofoil
Pylon 3	H	FRP	Complex	Pylon
Engine 3	I	Assembly	Shell	Cylinder
Pylon 4	J	FRP	Complex	Pylon
Engine 4	K	Assembly	Shell	Cylinder
Vert stabiliser 1	L	FRP	Beam	Aerofoil
Vert stabiliser 2	M	FRP	Beam	Aerofoil
Horz stabiliser	N	FRP	Beam	Aerofoil
Front landing gear	O	Assembly	Complex	Assembly
Rear landing gear	P	Assembly	Complex	Assembly
Name	**Element ID**	**Boundary**	-	-
Tarmac	1	Ground	-	-

Table 3: List of joints and their properties for Turbine 1

Joint designations for Turbine 1					
Joint ID	**Element set**	**Coordinate**	**Type**	**Disp. DoF**	**Rot. DoF**
1	A, D	8, 15, 235.75	Bearing	[x, y, z]	[y, z]
2	B, D	8, 14, 254	Bearing	[x, y, z]	[y, z]
3	D, E	10, 15, 253	Bearing	[x, y, z]	[y, z]
4	D, C	8, 16, 254	Bearing	[x, y, z]	[x, y]
5	E, F	15, 15, 250	Bearing	[x, y, z]	[x, y]
6	F, G	15, 15, 183	Bolted	-	-
7	G, H	15, 15, 105	Bolted	-	-
8	H, I	15, 15, 5	Bolted	-	-
9	I, 1	15, 15, 0	Soil	-	-

Table 4: List of joints and their properties for Aeroplane 1

Joint designations for Aeroplane 1					
Joint ID	**Element set**	**Coordinate**	**Type**	**Disp. DoF**	**Rot. DoF**
1	A1, A2	34.2, 14.68, 5.165	Perfect	-	-
2	A2, A3	34.2, 60.96, 5.165	Perfect	-	-
3	A2, B	32.2, 29.79, 2.89	Lug	-	-
4	B, C	13.2, 42.67, 4.74	Complex	-	-
5	C, D	13.2, 40.17, 4.74	Complex	-	-
6	B, E	23.2, 30.79, 3.57	Complex	-	-
7	E, F	23.2, 28.29, 3.57	Complex	-	-
8	A2, G	36.2, 29.79, 2.89	Lug	-	-
9	G, H	45.2, 30.79, 3.57	Complex	-	-
10	H, I	45.2, 28.29, 3.57	Complex	-	-
11	G, J	55.2, 42.67, 4.74	Complex	-	-
12	J, K	55.2, 40.17, 4.74	Complex	-	-
13	A3, L	33.2, 68.58, 7.55	Lug	-	-
14	A3, M	35.2, 68.58, 7.55	Lug	-	-
15	A3, N	34.2, 64.58, 9.16	Lug	-	-
16	A1, O	34.2, 7.75, 1.75	Complex	-	-
17	A2, P	34.2, 29.67, 1.75	Complex	-	-
18	O, 1	34.2, 7.75, 0	Plane	[z]	[x, y]
19	P, 1	34.2, 29.67, 0	Plane	[z]	[x, y]