Technische Universiteit
Eindhoven
University of Technology

Manufacturing Systems Engineering
Department of Mechanical Engineering
De Rondom 70, 5612 AP Eindhoven
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Author
T.J.L. Schuijbroek

Supervisor(s)
L.F.P. Etman
I.J.B.F. Adan
T. Wilschut

Date
March 2, 2018

Local Buses in Dependency Structure Matrices

Master Thesis Report

Local buses in Dependency Structure Matrices

Tiemen J.L. Schuijbroek

March 2, 2018

Abstract

Systems such as organizations, products, and processes consist of elements with interdependencies. The network of dependencies between elements of a system can be captured using the Dependency Structure Matrix (DSM) modeling technique. Algorithms and heuristics exist to detect decompositions within the network of dependencies. A bus module is a group of elements that act as a hub – i.e. its elements have many interdependencies with other elements in a system. A bus module may be global or local, depending on whether it integrates the whole system or a module. This article has two contributions. First of all, a novel algorithm is proposed for the detection of bus modules throughout a hierarchical system structure, i.e. the detection of global and local bus modules. Secondly, a new quality index is presented that is specifically designed to score decompositions that contain bus modules on a local level. The proposed algorithm and index are compared for results found in the literature for the Pratt & Whitney Aircraft Engine case and for a recently published Complex Elevator System case.

1 Introduction

Systems are often decomposed into multiple modules to reduce complexity, decrease development times, and facilitate a modular product family approach [1]. That is, the elements of a system are grouped into more comprehensible packages or modules. Finding such a modular decomposition is a non-trivial task. The dependencies between elements play a key role herein. Typically, one seeks modules that have many internal dependencies and relatively few external dependencies [2]. Solutions strike a balance between multiple objectives such as modularity and integrality and are often subject to experience and expertise.

The Dependency Structure Matrix (DSM) is a modeling technique that comprises the dependencies between elements in a system and lends itself for module detection analysis [2, 3]. The definition of a system, component, or dependency adapts to the modeling context. Examples are products consisting of components and connections, organizations of people with communications, and a process consisting of tasks and dependencies.

In a DSM, the elements are displayed on both axes of a matrix in identical order. See Figure 2a for an example of the Ford Climate Control System. The matrix cell values represent the dependencies between the elements. Categorical, binary, weighted, and directed DSM variations exist. The DSM is closely related to the adjacency matrix of a graph [4]. The elements and dependencies within a system are represented by the vertices and edges in a graph, respectively, see Figure 1.

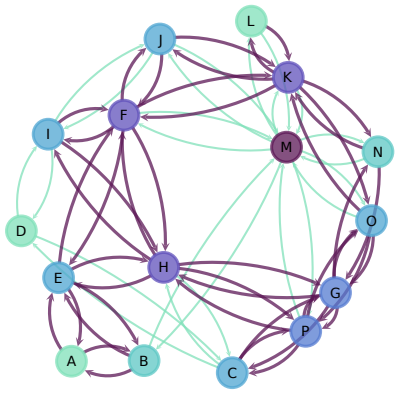
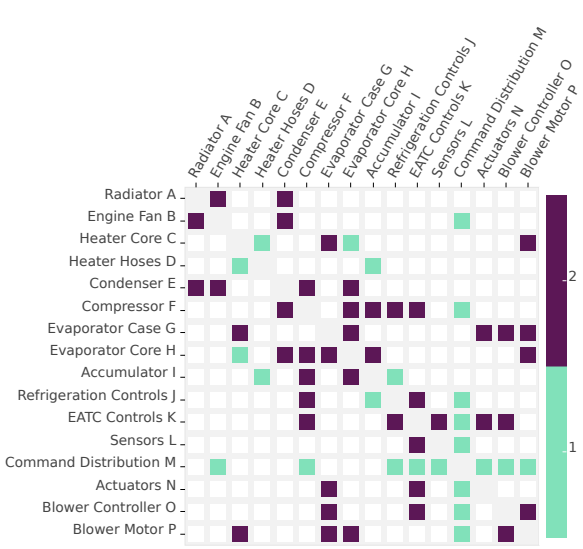
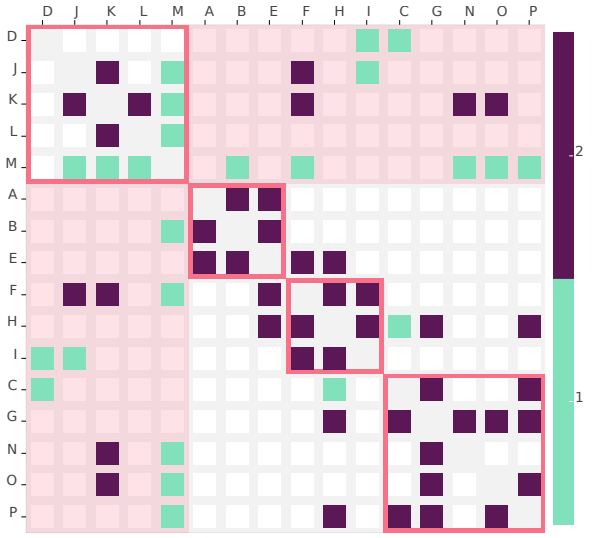


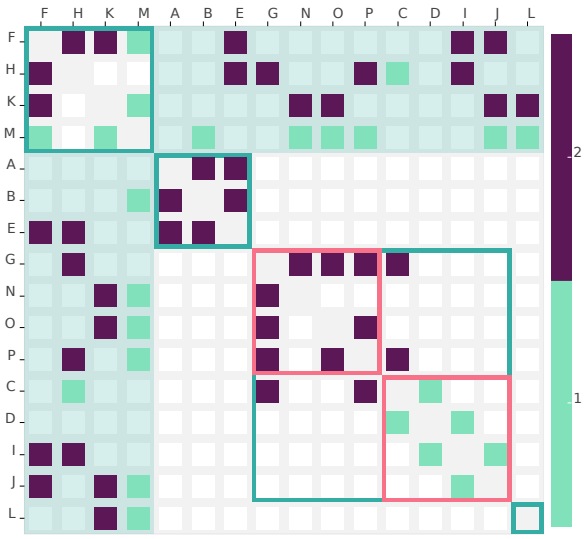
Figure 1: Graph representation of the Climate Control System DSM given in Figure 2a. Edge and vertex color denote edge weight and vertex degree (number of connections), respectively.



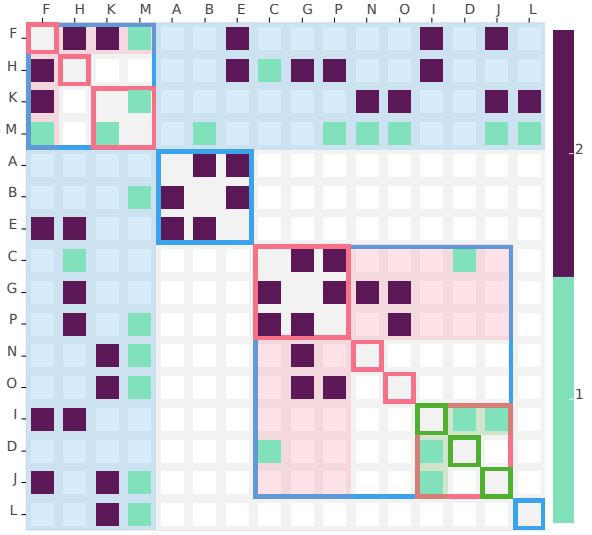
(a) Weighted DSM of the Ford Climate Control System [2].



(b) Handmade decomposition by Pimmler and Eppinger as found in [6].



(c) Decomposition by the HMC algorithm as presented in [6] with $\alpha = 2, \beta = 2.0, \mu = 2.0, \gamma = 1.5$.



(d) Decomposition with local buses obtained using the LB-HMC algorithm with $\alpha = 2, \beta = 2.0, \mu = 2.0, \gamma = 1.5$, as presented in Section 4

Figure 2: Climate Control System DSM with weak (1) and strong (2) component dependencies [2, 5, 6]. Shaded areas denote the connections from and to the bus modules (upper left).

DSMs are richer than adjacency matrices in the aspect that it is common to display multiple dependency types and strengths at once [7]. In case of the product DSM, exemplary dependency types are energy, spatial, information, and material flow [8]. Other variations exist that map performance indicators such as change propagation and robustness onto the DSM [2]. Another enrichment is the display of modules using a modified ordering of the elements on both axes and displaying module boundaries as rectangles in the DSM as in Figure 2 [9].

Creating a decomposition of a DSM can be done by hand or using clustering algorithms. Clustering by hand quickly becomes increasingly difficult for DSMs with more than a few components. Most existing clustering algorithms can take a graph's adjacency matrix or DSM as input and return a set of element groupings or modules as output.

However, their solutions often need the review of an expert to be conveyed correctly [2, 10]. Examples of clustering algorithms are K-means clustering [11], Spectral clustering [12], optimizing Module Description Length (MDL) [13], and Markov Clustering [14]. See [3, 15–17] for reviews and overviews of more clustering algorithms and applications.

Some clustering algorithms are also capable of finding a hierarchical decomposition instead of a single level of modules for a system [15, 16]. This difference is illustrated by Figure 2b and Figure 2c, where the former shows a single level handmade decomposition and the latter a hierarchy (modules within modules). Wilschut et al. presented a hierarchical clustering algorithm named Hierarchical Markov Clustering based on the work on Markov Clustering by Van Dongen [6, 14, 18].

Products typically contain a set of components that act as the hub for most of the others. For DSMs this is often referred to as the bus, named after the bus in a computer. Buses are the integrative modules of a system. From a graph perspective, bus elements would typically be the nodes with many incoming and outgoing edges – or node degree. In Figure 1, nodes H and M are illustrative examples of bus nodes. In clustering algorithms, assigning bus elements to a module can be hard and counter intuitive [6]. This drives the need for the algorithmic detection of bus modules that groups highly integrative elements together [6, 10, 12, 13, 19, 20]. In Figure 2b for example, the elements $\{D, J, K, L, M\}$ are assigned to the bus.

Bus modules that integrate the entire system are called *global buses*. Examples of global buses are module $\{D, J, K, L, M\}$ in Figure 2b and module $\{F, H, K, M\}$ in both Figure 2c and Figure 2d. Buses that integrate the elements of a module are called *local buses* and are seldom covered. Modules $\{F\}$, $\{C, G, P\}$ and $\{I\}$ are the local buses in Figure 2d.

Wilschut et al. mentions the detection of local buses as future work. Even though Sharman and Yassine mention the existence of *auxiliary buses* and Hölttä-Otto et al. of *weak buses*, they do not provide algorithms to detect them [10, 21]. The first contribution of this article addresses this. In Section 4 an algorithm is proposed to detect buses on a local level throughout a hierarchical decomposition of a system.

Clustering algorithms typically have a few parameters that can be tuned to optimize the resulting decompositions. Modularity indices have been introduced to rank different decompositions of a system by quality measures [13, 21–23]. Typically, indices trade-off objectives such as dependency density within a module and independence of other modules, regardless of the method used to obtain the result. Yu, Yassine, and Goldberg address global bus modules in the calculation of Model Description Length in [13], as well as Jung and Simpson in [22]. However, neither treat hierarchical decompositions or local bus modules. The second contribution of this work is a novel index capable of ranking hierarchical decompositions with local bus modules.

The outline is as follows: first, related work is discussed in Section 2 and is followed by several definitions regarding a decomposition in Section 3. In Section 4, the algorithm is presented that obtains decompositions with local bus modules for a given DSM. In Section 5, the index is introduced as a means to rank this type of decompositions for a given DSM. Two case studies are considered using the algorithm and index in Section 6. The first case study considers the Pratt & Whitney Aircraft Engine [5]. It has been studied thoroughly before and therefore is considered a good benchmark case for the proposed algorithm and index. The second case study considers a relatively new Complex Elevator System case. It was only recently published by Niutanen et al. in [24]. Conclusions and future work are to be found in Section 7 and Section 8.

2 Related work

This section covers the principles of a bus detection algorithm and two existing decomposition indices that can handle bus modules. The DSM of the Ford Climate Control System is used as the example, which is shown in Figure 2a. This Climate Control System has been modeled by its 16 major components, which are shown on both axis of the DSM. In further figures, the components have been replaced by their abbreviated labels, as shown in the other figures of Figure 2.

Eppinger and Browning made an abstraction of the original Climate Control System DSM by Pimmler and Eppinger, which reduces four different dependency types to a single dependency strength varying between weak and strong [2, 5]. This has been quantified in this article using weights of 1 and 2, respectively. Assigning some form of numerical weights to the dependencies in a DSM is a necessary step to obtain viable input for clustering algorithms. Essentially, an adjacency matrix is obtained from the DSM.

The algorithm presented in Section 4 incorporates a hierarchical clustering algorithm and a bus detection algorithm. Hierarchical Markov Clustering (HMC) is the chosen hierarchical clustering algorithm [6]. This algorithm has several key benefits. Among them is the absence of a prescribed number of modules or levels in the resulting hierarchical decomposition. As such, little knowledge of the system is required to obtain a decomposition. It provides flexibility in the output using three intuitive parameters that affect the probability of larger clusters or more hierarchical levels in the result, for example. Its limitation is that it can only take a positively weighted square matrix as input. Therefore, undesired or detrimental dependencies cannot be included, which are often given by negative values in a DSM.

Wilschut et al. introduced a bus detection algorithm in [6], which we will refer to as Gamma bus detection after its only parameter γ. Section 2.1 elaborates on this algorithm. When referring to HMC, we refer solely to the clustering algorithm – without Gamma bus detection. Both are used in the algorithm presented in Section 4, but can be replaced by other algorithms with the same functionality (being clustering hierarchically and bus detection).

2.1 Gamma bus detection

Bus detection algorithms try to detect which elements of a system are significantly more integral than others. Gamma bus detection selects elements with a significantly higher degree in a DSM than others to be the bus. A DSM element's degree is the number of nonzero entries in its row and column or number of incoming and outgoing edges in the network graph. For example in Figure 2a or Figure 1, element A of the Climate Control System has a degree of 4 and M has a degree of 8.

Let the array D_{NZ} contain the sorted non-zero element degrees of a DSM. The sorted non-zero element degrees of the Climate Control System are:

$$\begin{aligned} D_{\mathrm{NZ}} &= [d_A, d_D, d_L, d_B, d_N, d_C, d_E, d_I, \\ &\quad d_J, d_O, d_G, d_P, d_F, d_H, d_K, d_M] \\ &= [4, 4, 4, 6, 6, 8, 8, 8, \\ &\quad 8, 8, 10, 10, 12, 12, 12, 16] \end{aligned} \tag{1}$$

Where $d_A, d_B, \ldots, d_P$ denote the element degree of the Climate Control System's components with label $A, B, \ldots, P$.

Let D_{B} and D_{NB} hold the sorted degrees of the elements that are going to be assigned to the bus or remain non-bus elements. Bus elements are defined as the elements with a degree that is higher than a factor γ times the median of the remaining non-bus degrees, which is denoted $m(D_{\mathrm{NB}})$.

Initially, D_{NB} is set to D_{NZ}. The elements with an element degree greater than $\gamma \cdot m(D_{\mathrm{NB}})$ are then moved from D_{NB} to D_{B} recursively, until no more new bus elements are found for a given input parameter γ. As such, γ has to be greater than 1, since all elements would become bus elements in that case.

For $\gamma = 1.5$, this algorithm results in the following for the Climate Control System example:

1. $D_{\mathrm{NB}} = D_{\mathrm{NZ}}$.
2. $m(D_{\mathrm{NB}}) = 8$, so the threshold is $1.5 \cdot 8 = 12$ and thus $\{F, H, K, M\}$ are added to D_{B} and removed from D_{NB}.
3. $m(D_{\mathrm{NB}}) = 8$ again. This means the threshold is unchanged and no more elements need to be added to D_{B} and the algorithm stops.

2.2 Indices with bus handling

Algorithms are used to obtain decompositions and indices have been introduced to rank them according to quality metrics. These measure the quality of a decomposition with respect to the dependencies in the DSM of a system. Most modularity indices in the literature reward decompositions that contain rather independent modules with strong dependencies within each module. In general, bus modules have a negative influence on the metrics used in these indices as their nature is integrative rather than independent.

Two indices have been found in the literature that explicitly account for bus modules in a system: Model Description Length (MDL) by Yu, Yassine, and Goldberg and the modularity index (MI) by Jung and Simpson [13, 22]. Such functionality is necessary to correctly rank decompositions that contain bus modules. In the remainder of this section MDL and MI are discussed in more detail.

2.2.1 Model Description Length

MDL is a measurement of the required effort to describe all the deviations from a perfectly modular decomposition. Perfect modularity is defined as a system with equally sized modules, that each have as much internal dependencies and as little external dependencies as possible. Exceptions are dependencies from and to bus modules, which should be maximized, too.

Following the simplified notation for MDL by Hölttä-Otto et al., MDL is given by Equation (2). The weight of the three summation terms has been set to $\frac{1}{3}$ to that end.

$$\begin{aligned} \mathrm{MDL} = \frac{1}{3}\big(n_c \log n_n + \log n_n \sum_{i=1}^{n_c} \mathrm{cl}_i\big) \\ + \frac{1}{3}S_1 + \frac{1}{3}S_2 \end{aligned} \tag{2}$$

Where n_c is the total number of modules, n_n is the number of nodes (or elements in the DSM) and cl_i is the size of module i.

The first term aims to equalize the size of modules with respect to the number elements. Too many modules result in a higher description length. S_1 represents number of missing entries (equal to zero) in the DSM inside modules and from and to the bus modules. S_2 represents the number of nonzero entries between non-bus modules. With slight adjustments, the MDL can also be calculated for a normalized weighted matrix.

One of the advantages of this index is that it takes cluster size into account with respect to the number of elements involved. A drawback is that the index is an unbounded number where better decomposition quality results in a lower index value. The proposed index in Section 5 sums metrics for each sub-system in a hierarchical decomposition. An unbounded number is therefore not applicable to the proposed approach as the number of subsystems in a decomposition is not pre-defined.

2.2.2 Modularity Index

The Modularity Index (MI) by Jung and Simpson consists of three metrics that are calculated for each module, with an updated set of equations if a bus module is present. The three metrics can be summarized as module independence (MI_1), module density (MI_2) and diagonalization (MI_3). A normalized weighted sum results in a single MI value for a modular decomposition.

MI_1**: independence** For non-bus modules, this is the fraction of dependencies within each non-bus module with respect to the total number of dependencies between its own elements and other non-bus elements. Both the fraction of dependency *count* and fraction of dependency *weight* are taken into account.

MI_2**: density** The dependency density is defined as the the number of dependencies within each module with respect to the number of potential dependencies in each module. Similar to MDL, bus modules are an exception where the number of dependencies from and to bus modules are also taken into account, as well as the total number of potential dependencies from and to bus modules.

MI_3**: diagonalization** Besides the structural aspect of a product structure, diagonalization also rewards the dependencies closer to the diagonal. This makes more sense from a process DSM perspective, where more distance to the diagonal indicates further spread-out feedback loops. The bus is excluded from this metric.

The line of thought of the first two metrics is most relevant for hierarchical decompositions. The diagonalization metric is less applicable to such a decomposition, where the order of the elements is largely dominated by the top-down ordering of all nodes. A downside to the implementation of the independence metric MI_1 is that it is not necessarily bounded between $[0, 1]$ and can become negative. The used normalization scheme in MI is the cause of this. This is a drawback if it were to be used in the proposed index in Section 5. The lack of strict bounds would be obfuscating in sums of intermediate metrics for the multiple sub-systems in a hierarchical decomposition.

Therefore, a simplified, strictly bounded version of both the independence and density metric is included in the proposed index presented in Section 5. Independence and density are calculated for each sub-system as a whole in a hierarchical decomposition as opposed to each module in a system as done by Jung and Simpson, resulting in a single normalized value bounded between $[0, 1]$.

3 Decompositions as trees

The decomposition of a DSM can be modeled using trees. Trees provide an intuitive representation as well as formal means to describe a decomposition. Figure 3 provides a tree view of the handmade modular decomposition of the Climate Control System shown in Figure 2b. In this tree $\{0\}$ denotes the root node of the tree and nodes $\{1, 2, 3, 4\}$ represent the modules. The dark color of node $\{1\}$ in the tree indicates that it is a bus module. The nodes $\{A, B, \ldots, P\}$ correspond to the DSM elements.

Hierarchical Markov Clustering (HMC) yields hierarchical decompositions such as Figure 2c with the corresponding tree shown in Figure 4. A hierarchical decomposition contains sub-modules within modules such as nodes $\{5, 6\}$ within $\{3\}$ in the given tree, whereas a modular decomposition only contains a single level of modules – directly followed by the DSM elements.

Decompositions obtained using a clustering algorithm may contain disjoint systems, without common dependencies. In those cases the result may not be a single tree with a single root node, but in fact a set of trees or a *forest* – each with their own root node.

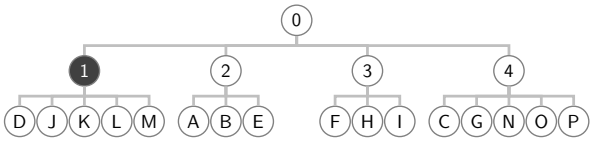


Figure 3: Handmade modular decomposition corresponding to Figure 2b.

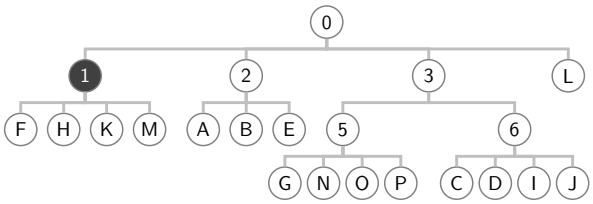


Figure 4: Hierarchical decomposition obtained using HMC corresponding to Figure 2c.

3.1 Forest of trees

Any of the given DSM decompositions can be described using a forest of rooted, directed, labeled trees. A description thereof is given in the remainder of this section and is used to describe the algorithm in Section 4 and decomposition index in Section 5.

A forest $\mathbb{F} = (\mathcal{N}, \mathcal{E})$ is a directed acyclic graph, with nodes $\mathcal{N}$ and edges $\mathcal{E}$. An edge $(p, c) \in \mathcal{E}$ denotes a direct parent-child relationship between parent node p and child node c. The DSM's elements are the leaf nodes $\mathcal{L} \subset \mathcal{N}$ of this graph and have no children by definition.

Let $\mathbb{F}^{\mathcal{R}} = (\mathcal{N}, \mathcal{E})$ denote a forest with root nodes $r \in \mathcal{R}$ of the trees in it. Let a tree with root node r then be denoted by $\mathbb{T}^r = (\mathcal{N}^r, \mathcal{E}^r)$, where $\mathcal{N}^r \subset \mathcal{N}$ and $\mathcal{E}^r \subset \mathcal{E}$. Each tree is a group of *connected components* in the forest. The following holds for any tree in the forest:

Rooted A tree is assumed to have a single root node and DSM elements as the leaf nodes.

Directed Every node has exactly one parent, except for the tree's root node which has none. A node can have any number of children.

Labeled Bus nodes are nodes that are labeled as a bus. Dark shaded nodes in figures represent nodes that are labeled as a (local) bus. Modules are denoted by numbers and leaf nodes (DSM elements) by uppercase letters.

Note that any branch (sub-tree) starting in any node $n \in \mathcal{N}$ is also a tree following these definitions. A branch is denoted $\mathbb{T}^n = (\mathcal{N}^n, \mathcal{E}^n)$, using the superscript n to indicate the branch's root node.

The set of nodes $\mathcal{N}$ can be partitioned into a set of bus nodes and a set of non-bus nodes. These are denoted as $\mathcal{N}_{\mathrm{B}}$ and $\mathcal{N}_{\mathrm{NB}}$, respectively.

3.2 Parent, children, grandchildren

The parent of a node n is denoted by $\mathcal{P}(n) \subset \mathcal{N}$. Each node n has only one parent node p for which a unique edge $(p, n) \in \mathcal{E}$ exists.

The set of child nodes of a parent node n are the nodes with a direct edge originating from node n. The set of children of a node n are given by $\mathcal{C}(n) \subset \mathcal{N}$ and defined as:

$$\mathcal{C}(n) = \{c \in \mathcal{N} \mid (n, c) \in \mathcal{E}\} \tag{3}$$

The degree of a node n equals the number of children it has. This can be obtained using the set cardinality (size) of $\mathcal{C}(n)$, denoted as $|\mathcal{C}(n)|$.

Let $\mathcal{L}$ denote the set of leaf nodes, which are the nodes with no children.

$$\mathcal{L} = \{n \in \mathcal{N} \mid \quad |\mathcal{C}(n)| = 0\} \tag{4}$$

Let $\mathcal{G}(n)$ denote the set of grandchildren of a node n. These are the children of children of this node, as defined in Equation (5). If any child node $c \in \mathcal{C}(n)$ is a leaf node, that leaf node itself is also included in the grandchildren of n.

$$\mathcal{G}(n) = \bigcup \big\{\mathcal{C}(c) \mid c \in \mathcal{C}(n)\big\} \cup \big(\mathcal{C}(n) \cap \mathcal{L}\big) \tag{5}$$

Where the first part corresponds to the union of all sets of children of children and the second part are the children that are leaf nodes of a node n.

3.3 Depth and height

The depth of a node n in forest $\mathbb{F}^{\mathcal{R}}$ is measured from the root nodes that have a depth of 0 and is defined by Equation (6). The set of nodes with a certain depth d in the forest is defined by Equation (7). Conversely, let the height of a node n be given by the maximum depth of any of the leaf nodes $\mathcal{L}^n$ in its branch, see Equation (8).

$$\mathrm{depth}(n) = \begin{cases} 0 & n \in \mathcal{R} \\ \mathrm{depth}(\mathcal{P}(n)) + 1 & n \in \mathcal{N} \setminus \mathcal{R} \end{cases} \tag{6}$$

$$\mathcal{D}(d) = \{n \in \mathcal{N} \mid \mathrm{depth}(n) = d\} \tag{7}$$

$$\mathrm{height}(n) = \max\{\mathrm{depth}(l) \mid l \in \mathcal{L}^n\} - \mathrm{depth}(n) \tag{8}$$

Where $\mathcal{D}(d)$ denotes all nodes at a certain depth in the forest and $\mathcal{L}^n$ the leaf nodes of the branch starting in node n.

4 Local bus detection

The algorithm presented in this section integrates a bus detection algorithm and a hierarchical clustering algorithm to obtain a hierarchical decomposition of a system with buses on a local level.

The algorithm outline is given in Section 4.1 and the key steps are given in Section 4.2. Finally, this section is concluded with an example of the algorithm using Hierarchical Markov Clustering and Gamma bus detection for the Climate Control System DSM in Section 4.3.

4.1 Algorithm outline

In short, the algorithm performs a local bus search from top to bottom over each module in each level of a hierarchical decomposition. If the bus detection algorithm detects bus elements, it replaces that module's branch (the sub-tree starting at that module) with a new branch.

That new branch consists of the results of the hierarchical clustering algorithm on both the bus elements and non-bus elements. As such, when the algorithm searches for bus elements at a certain level, it only replaces parts of the tree that are deeper in the tree, leaving the checked levels intact.

4.2 Algorithm steps

For a positively weighted adjacency matrix $\mathbf{A}$, the algorithm is given in Algorithm 1. It results in a decomposition as a forest $\mathbb{F}^{\mathcal{R}}$, as defined in Section 3. It consists of five key steps, being initialization, iterations, local bus detection, re-clustering and the merge of results. These key steps are described below.

Algorithm 1 Find local bus decomposition

```
1: procedure DECOMPOSITION(A)
2:     F^R ← HIERARCHY(A)
3:     for r ∈ R do
4:         d ← 0
5:         while d ≤ height(r) − 1 do
6:             for n in D(d) do
7:                 L^n_B, L^n_NB ← DETECT BUS(A^n)
8:                 if L^n_B ≠ ∅ then
9:                     F^B ← HIERARCHY(A^n_{L_B})
10:                    F^{NB} ← HIERARCHY(A^n_{L_NB})
11:                    T^n ← BUS MERGE(F^B, NB)
12:                end if
13:            end for
14:            d ← d + 1
15:        end while
16:    end for
17:    return F^R
18: end procedure
```

1: Initialization The forest $\mathbb{F}^{\mathcal{R}}$ is initialized at Line 2 with a hierarchical decomposition by the hierarchical clustering algorithm. The decomposition does not contain any bus modules at this point.

2. Iterations At Lines 3 to 6 an iteration is started over each node at a certain depth $n \in \mathcal{D}(d)$ in each tree with a root node r in the forest $\mathbb{F}^{\mathcal{R}}$. As such, each tree will be checked top down for the presence of bus elements in its modules.

3. Local bus detection The local bus detection is done at Line 7, where the bus detection algorithm is applied to the sub-matrix $\mathbf{A}^n$ of adjacency matrix $\mathbf{A}$. This sub-matrix is the adjacency matrix of all leaf nodes (DSM elements) in the branch starting at node n. It returns a set of bus leaf nodes (elements) $\mathcal{L}^n_{\mathrm{B}}$ and the set of remaining non-bus leaf nodes $\mathcal{L}^n_{\mathrm{NB}}$.

4. Re-clustering If the set of bus nodes is not empty (Line 8), both sets of nodes are subjected to the hierarchical clustering algorithm at Line 9 and 10. Each call returns it's respective temporary decomposition, given by forests $\mathbb{F}^{\mathcal{B}}$ and $\mathbb{F}^{\mathcal{NB}}$.

5. Merge results These temporary forests are then merged at Line 11, where the branch $\mathbb{T}^n$ starting at node n in the original forest is replaced by the result of merging $\mathbb{F}^{\mathcal{B}}$ and $\mathbb{F}^{\mathcal{NB}}$. This merge works as follows. The root node n remains intact, but its children are replaced. The root nodes $\mathcal{B}$ of the bus forest $\mathbb{F}^{\mathcal{B}}$ become children of n. These bus root nodes receive the bus label, indicating they are bus nodes for their parent n. It is possible that the hierarchical clustering algorithm returns a forest $\mathbb{F}^{\mathcal{NB}}$ with a single root node for the non-bus set of nodes. In that case, the children of that root node are added to the children of n. Otherwise, the root nodes $\mathcal{NB}$ are added to the children of n, similar to the bus forest. This distinction is made because a single bus module is allowed, while a a single non-bus module would be counter intuitive. A bus module cannot perform an integrative function for a single non-bus module.

4.3 Example: LB-HMC

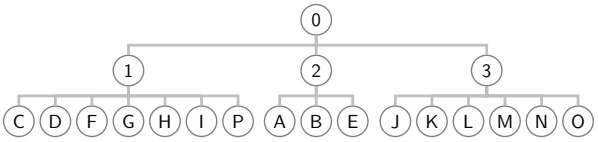
The implementation of the algorithm uses Hierarchical Markov Clustering (HMC) and Gamma bus detection as described in Section 2.1. The implemented algorithm is hence forward referred to as Local Bus HMC or LB-HMC for short.

Because of HMC's input requirements, LB-HMC also requires a positively weighted DSM or adjacency matrix as input, such as the Climate Control System DSM given in Figure 2a. The algorithm adopts the parameters of HMC and Gamma bus detection with no additions. As such, it has α, β and μ to tune HMC and γ to tune Gamma bus detection.

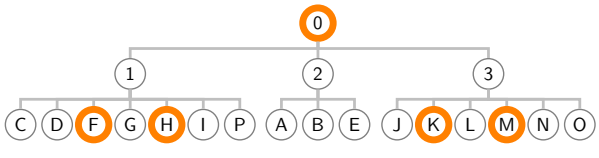
The following describes the first bus detection iteration of the LB-HMC algorithm for the Climate Control System DSM. The steps of this first iteration and the final result of the algorithm are shown in Figure 5:

1: Initialization The initial hierarchical decomposition that is obtained using HMC is given in Figure 5a. That is, the algorithm finds three modules $\{1, 2, 3\}$ in a single system $\{0\}$.

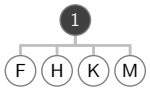
2. Iterations The initial decomposition has a single root node and three levels. First, the only root node at depth 0 will be checked for buses.



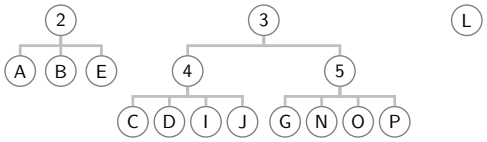
(a) Initial hierarchical decomposition.



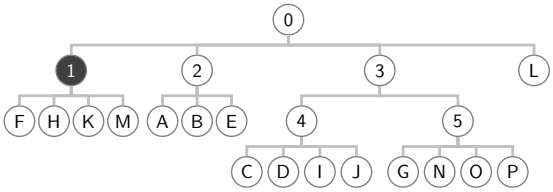
(b) Detected bus nodes (marked) at depth 0 for its only module, module 0 (also marked).



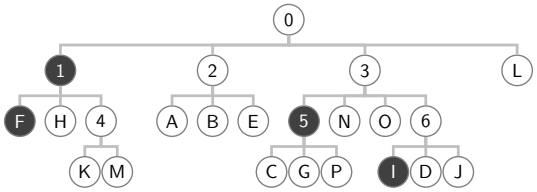
(c) Results of re-clustering the bus elements of node 0 with the root node already marked as a bus node.



(d) Results of re-clustering the remaining non-bus elements of node 0.



(e) Decomposition after insertion of the bus and non-bus re-clustering results for node 0.



(f) Final decomposition after all levels and modules have been checked for local buses.

Figure 5: Example of LB-HMC applied to the Climate Control System DSM given in Figure 2a. The first figure is the initial hierarchy, followed by the first iteration of the local bus search routine and the final result. The used parameters are $\alpha = 2, \beta = 2.0, \mu = 2.0, \gamma = 1.5$.

Subsequently, the algorithm will iterate over the modules at depth 1 and onwards. More levels and modules may be introduced as local bus modules are found and parts of the decomposition are replaced.

3. **Local bus detection** The result of the bus detection for node 0 is given in Figure 5b. It finds 4 bus elements, being $\{F, H, K, M\}$.

4. **Re-clustering** The found bus elements and non-bus elements are re-clustered separately. The result of the re-clustering of the bus elements is given in Figure 5c. It has a single root node, which is marked as a bus node. The result of the re-clustering of the non-bus elements is given in Figure 5d. The result is a forest with multiple root nodes $\{2, 3, L\}$.

5. **Merge results** For this merge, the children of node 0 from Figure 5a are replaced by the root node $\{1\}$ of the bus decomposition and the root nodes $\{2, 3, L\}$ of the non-bus decomposition. As the non-bus decomposition was already a forest with multiple root nodes, there is no need to remove a single root node.

The iterations continue for modules at depths of 1 and 2 to detect additional local bus modules. The final result thereof is shown in Figure 5f.

1. For node $\{1\}$ at depth 1, this results in $\{F\}$ as the local bus. $\{H, K, M\}$ is re-clustered as a result of this, which separates $\{H\}$ and $\{K, M\}$ into separate modules.

2. For node $\{2\}$, no local buses are found and therefore no changes are made to its branch.

3. For node $\{3\}$, this results in $\{C, G, P\}$ as the local bus elements, which end up in a single module as node $\{5\}$.

4. For node $\{4\}$ at depth 2, no local buses are found and therefore no changes are made to its branch.

5. For node $\{5\}$, no local buses are found.

6. For node $\{6\}$, this results in $\{I\}$ as the local bus.

5 Local Bus Decomposition Index

In this section, the Local Bus Decomposition Index (LBDI) is introduced. It is a normalized metric that can be used to rank the quality of decompositions for a DSM. It is bounded between 0 and 1, where a higher value indicates higher quality. The LBDI requires a DSM's adjacency matrix $\mathbf{A}$ and a decomposition represented as a rooted, directed, labeled tree as explained in Section 3.

First, some additional definitions are given in Section 5.1 and Section 5.2. This is followed by the independence and density metrics in Section 5.3 and Section 5.4. Finally, these metrics are combined into the LBDI value in Section 5.5.

5.1 System nodes

The LBDI requires a tree $\mathbb{T}^r$ with a single root r as input as opposed to a forest with multiple trees. A root node r is added if the decomposition contains multiple independent trees, which has the forest's $\mathbb{F}^{\mathcal{R}}$ root nodes $\mathcal{R}$ as its children such that $\mathcal{C}(r) = \mathcal{R}$.

Only the system nodes in a decomposition are measured for quality. The system nodes are denoted $\mathcal{N}_{\mathrm{S}}$ and are defined as:

- The root node.
- All nodes with at least one non-leaf child.
- All nodes with at least one bus child.

For example, Figure 6 shows the tree of a decomposition for the Climate Control System, where nodes $\{0, 1, 3, 6\}$ are marked as system nodes. They are system nodes since node $\{0\}$ is the root node and nodes $\{1\}$, $\{3\}$, and $\{6\}$ all have both a bus child and a non-leaf child.

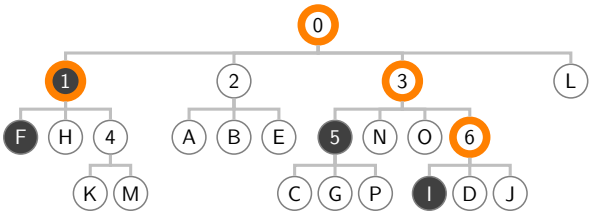


Figure 6: Decomposition obtained using LB-HMC corresponding to Figure 2d. System nodes are marked.

The quality of each system node in the decomposition is expressed using an independence and a dependency density metric. These metrics work similar to the metrics discussed in Section 2.2. These values for all system nodes can be combined into a single normalized LBDI value for the whole decomposition, as explained in Section 5.5.

5.2 Adjacency matrix coarsening

The quality of each system node is based on the density and independence of its children. In this respect, each system node is considered to form a separate sub-system with its own associated DSM. In this DSM, the grandchildren of the system node are the elements and the children of the system node are the modules. As such, the system node's DSM is represented by a coarsened sub-matrix of matrix $\mathbf{A}$. That is, the dependency weights of the system node's DSM between elements i and j is the sum of all the dependencies between the leaf nodes in the branches starting in grandchildren i and j.

Let $\mathbf{A}^n_{\mathcal{G}}$ denote the coarsened sub-matrix of $\mathbf{A}$ such that all cells indicate the adjacency between two grandchildren i and j as defined in Equations (9) and (10).

$$a'(i,j) = \sum_{l_i \in \mathcal{L}^i} \sum_{l_j \in \mathcal{L}^j} \mathbf{A}(l_i, l_j) \tag{9}$$

$$\mathbf{A}^n_{\mathcal{G}}(i,j) = \begin{cases} a'(i,j) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Where the scalar $a'(i,j)$ is the sum of all dependencies in adjacency matrix $\mathbf{A}$ between the leaf nodes $\mathcal{L}^i$ and $\mathcal{L}^j$ of the branches starting in node i and j. The diagonal of $\mathbf{A}^n_{\mathcal{G}}$ is set to 0, as is common practice in DSM modeling (an element does not depend on itself).

Let $\mathbf{B}^n_{\mathcal{G}}$ be the binary variant of $\mathbf{A}^n_{\mathcal{G}}$ and is defined as:

$$\mathbf{B}^n_{\mathcal{G}}(i,j) = \begin{cases} 1 & \text{if } \mathbf{A}^n_{\mathcal{G}}(i,j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

This binary matrix is used to count amounts of dependencies as opposed to their dependency weights.

5.3 Independence metric

A system node's adjacency matrix $\mathbf{A}^n_{\mathcal{G}}$ may contain both dependencies within modules (intra) and dependencies between modules (inter). The balance between intra-module and inter-module connections is used to express the independence of the modules. Relatively more intra-module dependencies than inter-module dependencies result in a higher independence metric.

The independence metric is calculated for non-bus modules only, since bus modules should be very interdependent by definition. It accounts for both the fraction of dependency weight or count that are intra-module dependencies with respect to the total dependency weight or count of both the intra- and inter-module dependencies.

The independence of a system node n is denoted $\iota^n(x_\mathrm{w})$. An input parameter $x_\mathrm{w} \in [0, 1]$ defines the contribution of the weight independence ι^n_w versus the count independence ι^n_b as defined in Equation (12).

$$\iota^n(x_\mathrm{w}) = x_\mathrm{w}\iota^n_\mathrm{w} + (1 - x_\mathrm{w})\iota^n_\mathrm{b} \tag{12}$$

This prioritizes either *stronger* dependencies or *the existence of* dependencies when measuring the independence by tuning x_w.

The weight independence ι^n_w of a node n is the fraction of connection weight within non-bus children (modules) of n with respect to the total connection weight between the grandchildren (elements) of those non-bus children as defined in Equations (13)–(15).

$$\mathrm{w}^n_\mathrm{NB} = \sum_{(i,j)\in\mathcal{G}_\mathrm{NB}(n)\times\mathcal{G}_\mathrm{NB}(n)} \mathbf{A}^n_{\mathcal{G}}(i,j) \tag{13}$$

$$\mathrm{w}^n_\mathrm{NB,intra} = \sum_{c\in\mathcal{C}_\mathrm{NB}(n)} \sum_{(i,j)\in\mathcal{C}(c)\times\mathcal{C}(c)} \mathbf{A}^n_{\mathcal{G}}(i,j) \tag{14}$$

$$\iota^n_\mathrm{w} = \frac{\mathrm{w}_\mathrm{NB,intra}}{\mathrm{w}_\mathrm{NB}} \tag{15}$$

Where w^n_NB is the total dependency weight between grandchildren of a node n from non-bus children of n. This subset of grandchildren of a node n is denoted by $\mathcal{G}_\mathrm{NB}(n)$. $\mathrm{w}^n_\mathrm{NB,intra}$ is the part of dependencies that resides within (intra) the non-bus children, between their grandchildren. The weight independence ι^n_w then follows after division.

Similar to the weight independence calculation, the binary independence ι^n_b follows using the binary variant of the coarsened adjacency matrix $\mathbf{B}^n_{\mathcal{G}}$ instead of the weighted matrix $\mathbf{A}^n_{\mathcal{G}}$. This results in Equations (16)–(18).

$$\mathrm{b}^n_\mathrm{NB} = \sum_{(i,j)\in\mathcal{G}_\mathrm{NB}(n)\times\mathcal{G}_\mathrm{NB}(n)} \mathbf{B}^n_{\mathcal{G}}(i,j) \tag{16}$$

$$\mathrm{b}^n_\mathrm{NB,intra} = \sum_{c\in\mathcal{C}_\mathrm{NB}(n)} \sum_{(i,j)\in\mathcal{C}(c)\times\mathcal{C}(c)} \mathbf{B}^n_{\mathcal{G}}(i,j) \tag{17}$$

$$\iota^n_\mathrm{b} = \frac{\mathrm{b}_\mathrm{NB,intra}}{\mathrm{b}_\mathrm{NB}} \tag{18}$$

Where b^n_NB is the total dependency count between grandchildren from non-bus children and $\mathrm{b}^n_\mathrm{NB,intra}$ is the part thereof that resides within (intra) the non-bus children between their grandchildren.

The distinction of local buses can create a situation where there are no dependencies between non-bus children. See the $\{I, D, J\}$ module in Figure 2d for an example. In such cases, Equations (15) and (18) would result in divisions by zero. However, we assume these modules to be fully independent, resulting in an independence metric $\iota^n = 1$.

5.4 Density metric

In general, one strives for tightly integrated modules in a (sub-) system. Modules whose elements are very dependent on each other contribute to a higher density metric, which reflects their tight integration. The density of a (sub-) system is calculated for all modules (bus and non-bus). It is expressed as the fraction of potential dependencies within non-bus modules and from and to bus elements that exist. That is, bus modules should have both dense intra-module dependencies as well as inter-module dependencies.

The density metric of a system node n is denoted ρ^n as defined in Equation (19).

$$\rho^n = \frac{\mathrm{b}^n_\mathrm{B} + \mathrm{b}^n_\mathrm{NB,intra}}{\Delta^n_\mathrm{B} + \Delta^n_\mathrm{NB,intra}} \tag{19}$$

Where numerator denotes the existing dependency count related to all bus grandchildren b^n_B and within non-bus children $\mathrm{b}^n_\mathrm{NB,intra}$. The denominator denotes the number of potential dependencies

of the bus grandchildren Δ^n_{B} and within non-bus children $\Delta^n_{\mathrm{NB,intra}}$. These are elaborated on in the coming paragraphs.

In the numerator of Equation (19), $\mathrm{b}^n_{\mathrm{B}}$ denotes the dependency count from and to bus modules as defined in Equation (20) and $\mathrm{b}^n_{\mathrm{NB,intra}}$ denotes the dependency count within (intra) non-bus modules as defined in Equation (17).

$$\begin{aligned}\mathrm{b}^n_{\mathrm{B}} = &\sum_{(i,j)\in\mathcal{G}_{\mathrm{B}}(n)\times\mathcal{G}_{\mathrm{B}}(n)} \mathbf{B}^n_{\mathcal{G}}(i,j) \\ &+ \sum_{(i,j)\in\mathcal{G}_{\mathrm{B}}(n)\times\mathcal{G}_{\mathrm{NB}}(n)} \mathbf{B}^n_{\mathcal{G}}(i,j) + \mathbf{B}^n_{\mathcal{G}}(j,i)\end{aligned} \tag{20}$$

Where $\mathcal{G}_{\mathrm{B}}(n)$ represents the grandchildren of a node n from bus children of n and $\mathcal{G}_{\mathrm{NB}}(n)$ represents the grandchildren of n from non-bus children of n. As such, the first sum counts all dependencies between grandchildren from bus children and the second sum counts all dependencies from and to grandchildren from the non-bus children of n.

In the denominator of Equation (19), Δ^n_{B} denotes the number of potential dependencies from and to bus elements as defined in Equation (22) and $\Delta^n_{\mathrm{NB,intra}}$ does so for the intra non-bus module dependencies as defined in Equation (21). The potential number of dependencies is also referred to as the *area* of a part of a DSM.

$$\Delta^n_{\mathrm{NB,intra}} = \sum_{c\in\mathcal{C}_{\mathrm{NB}}(n)} |\mathcal{C}(c)|\cdot\big(|\mathcal{C}(c)|-1\big) \tag{21}$$

$$\begin{aligned}\Delta^n_{\mathrm{B}} = &|\mathcal{G}_{\mathrm{B}}(n)|\cdot\big(|\mathcal{G}_{\mathrm{B}}(n)|-1\big) \\ &+ 2\cdot|\mathcal{G}_{\mathrm{B}}(n)|\cdot|\mathcal{G}_{\mathrm{NB}}(n)|\end{aligned} \tag{22}$$

Where $\mathcal{C}_{\mathrm{NB}}(n)$ denotes the non-bus children of a node n. In Equation (21), the area is calculated for each non-bus child using the set cardinality (size) of its respective children $|\mathcal{C}(c)|$. The minus 1 excludes the diagonal of the area in the coarsened DSM. In Equation (22), the first term represents the number of potential dependencies between grandchildren of a node n from bus children of n. The second term represents the potential dependencies between grandchildren of a node n from bus children of n and the grandchildren of n from non-bus children of n.

The density metric ρ^n for a system node n then follows by division of the total count by the sum of the respective areas.

5.5 LBDI value

The LBDI value follows from the independence and density metrics of all system nodes. The normalization scheme uses the area of each system node as the weight for its metrics. As such, larger (sub-) systems have a larger contribution to the final index value as they are more complex. It has two parameters, being x_ι and x_{w}. The parameter $x_\iota \in [0,1]$ defines the contribution of the independence with respect to the density metric. The parameter $x_{\mathrm{w}} \in [0,1]$ defines the contribution of the dependency weight when calculating the independence metric, as explained in Section 5.3. As such, the LBDI$(x_{\mathrm{w}}, x_\iota)$ can be calculated using Equations (23)–(27).

$$\Delta^n = |\mathcal{G}(n)|\cdot\big(|\mathcal{G}(n)|-1\big) \tag{23}$$

$$\Delta = \sum_{n\in\mathcal{N}_{\mathrm{S}}} \Delta^n \tag{24}$$

$$\iota(x_{\mathrm{w}}) = \sum_{n\in\mathcal{N}_{\mathrm{S}}} \frac{\Delta^n\cdot\iota^n(x_{\mathrm{w}})}{\Delta} \tag{25}$$

$$\rho = \sum_{n\in\mathcal{N}_{\mathrm{S}}} \frac{\Delta^n\cdot\rho^n}{\Delta} \tag{26}$$

$$\mathrm{LBDI}(x_{\mathrm{w}}, x_\iota) = x_\iota\cdot\iota(x_{\mathrm{w}}) + \big(1-x_\iota\big)\cdot\rho \tag{27}$$

Where Δ^n is the area of a node n and Δ denotes the area sum of all system nodes. $\iota(x_{\mathrm{w}})$ denotes the normalized independence metric of all system nodes and ρ does so for the density. LBDI$(x_{\mathrm{w}}, x_\iota)$ is the result after applying the independence ratio x_ι with respect to the density.

It is also possible to calculate an index value for an individual system node by applying the LBDI parameters x_{w} and x_ι to the system node's independence $\iota^n(x_{\mathrm{w}})$ and density ρ^n metrics. This is referred to as the nodal decomposition index NDI, which is defined in Equation (28). The NDI can be used to gain insight into the decomposition quality on a nodal level.

$$\mathrm{NDI}(x_{\mathrm{w}}, x_\iota) = x_\iota\cdot\iota^n(x_{\mathrm{w}}) + \big(1-x_\iota\big)\cdot\rho^n \tag{28}$$

for all system nodes $n \in \mathcal{N}_{\mathrm{S}}$.

Table 1: Independence, density, area and nodal index for the marked system nodes in Figure 6 of the LB-HMC decomposition. Results are given for LBDI($x_{\mathrm{w}} = 1.0, x_{\iota} = 0.5$) = 0.7719.

n	ι^n	ρ^n	Δ^n	NDI
0	1.0	0.5278	110	0.7639
1	1.0	1.0000	12	1.0000
3	1.0	0.4286	56	0.7143
6	1.0	1.0000	6	1.0000

5.6 Example

The independence, density, area and nodal index are given in Table 1 for the system nodes in the LB-HMC decomposition of the Climate Control System shown in Figure 2d. The nodal indices are calculated using a weight ratio of 1.0 and an independence ratio of 0.5. This means that the given nodal indices are the average of their weight independence and density. It is also visible in Figure 2d that the independence scores of 1.0 are justified, since all of the given modules of the system nodes are fully independent. Because of the fully dense buses in for nodes {1} and {6}, their respective nodal indices have the perfect score of 1.

Table 2 shows a comparison of the LBDI value for the three decompositions given in Figure 2. The LBDI parameters are kept at $x_{\mathrm{w}} = 1.0$ and $x_{\iota} = 0.5$. The density of the handmade decomposition is significantly lower than the densities obtained using the HMC and LB-HMC algorithms. Especially the inclusion of the *Heater Hoses D* in the bus module is of a negative influence to the handmade decomposition's density metric. However, this may be due to expert knowledge not available to the algorithms. Even though the density metric of the LB-HMC decomposition is slightly lower on average, it outperforms HMC on independence metric resulting in a higher overall LBDI value. Especially the local bus module $\{C, G, P\}$ contributes to the lower density metric of the LB-HMC, which at the same time helps to maximize the independence metric locally. This is supported by the values in Table 1, where the resulting independence is 1.0 at the cost of a density of 0.4286.

Table 2: Independence and density metrics and LBDI($x_{\mathrm{w}} = 1.0, x_{\iota} = 0.5$) for the three decomposition methods given in Figure 2.

Method	ι	ρ	LBDI
Handmade	0.7273	0.3580	0.5426
HMC	0.8977	0.5864	0.7420
LB-HMC	1.0000	0.5438	0.7719

6 Case studies

Two case studies are considered in this section, being the Pratt & Whitney Aircraft Engine and the Complex Elevator System [24, 25]. The Pratt & Whitney Aircraft Engine is a much frequented DSM in the literature, which serves as a benchmark case. The presented results are compared with the decomposition obtained by Jung and Simpson in [22]. Since their modularity index' approach resembles our nodal index, as presented in Section 5, the most. The Complex Elevator System is a DSM that is recently published explicitly as a new challenging case study to consider.

6.1 Optimization method

In both cases, a Simple Genetic Algorithm (SGA) is used to tune the parameters of LB-HMC, as presented in Section 4. The proposed Local Bus Decomposition Index (LBDI) from Section 5 is used as the fitness function. See Chapter 7 of [26] of an in-depth explanation of an SGA. This subsection provides an overview of the SGA used to obtain the results. The values and bounds of all parameters have been set empirically, consistently generating stable results.

An SGA simulates an evolutional process by evolving a population of individuals. Evolution takes place using selection of better performing individuals, cross-over of individuals and mutation of individuals. Cross-over generates offspring of some selected individuals and mutation slightly adjusts the values in some individuals.

The individuals in the SGA optimization routine are represented by sets of algorithm parameters for LB-HMC. For HMC, α was bounded between 2 and 3, and β and μ were bounded between 1 and 5. For Gamma bus detection, the value for γ was also bounded between 1 and 5.

The fitness of a individual is given by the LBDI value for the resulting decomposition that is obtained with the individual as the LB-HMC settings. The weight fully contributes to the independence metric using $x_{\mathrm{w}} = 1$ while independence only contributes for 25% of the final score using $x_{\iota} = 0.25$.

The initial population consists of 500 randomly generated individuals, where each LB-HMC parameter is uniformly distributed within its respective bounds. After each generation selection takes place, followed by both blended cross-over and Gaussian mutation with each a probability of 30% to generate the new generation's population.

The tournament selection uses a tournament size of 3. This creates random groups of 3 individuals from the current population and selects the best individual from each group to be part of the new population.

Blended cross-over generates two new individuals, which are blended versions of two ancestors in the population. The blending ratio of each ancestor contributing to a new individual is uniformly distributed up to 50%. The blending ratio of both new individuals is the opposite of the other. Blended individuals replace their ancestors in the new population.

Gaussian mutation has a probability of 25% of changing each LB-HMC parameter. When an LB-HMC parameter is mutated, the standard deviations differ per parameter as some are more sensitive than others. The used standard deviations are 0.25 for α, 0.5 for β and μ, and 0.25 for γ. Mutated individuals replace their ancestors in the new population.

This meant roughly 200-250 new unique individuals per generation. The given results are the best individual after 10 generations. Most simulations showed no or marginal improvements in LBDI after 5 generations.

The SGA and LB-HMC algorithm have been implemented using Python 3. Performance on a machine with an Intel Core i5-6300HQ CPU and 8 GB of RAM was passable, with the heavier Complex Elevator Case taking roughly 5 minutes for 10 generations of optimization for a given pair of parameters for LBDI.

6.2 Pratt & Whitney Aircraft Engine

The Pratt & Whitney PW4098 Aircraft Engine is a much frequented DSM case study [2, 6, 12, 13, 18, 22, 25, 27]. The DSM includes a combination of the actual hardware dependencies of the engine and those of the development teams involved with them. It is a weighted and asymmetric DSM featuring 60 elements, four of which are sometimes left out as they represent the integration teams and no individual hardware components. Weak and strong dependencies are distinguished using weights of 1 and 2, respectively.

Jung and Simpson also considered this case study with 60 elements in [22]. Their proposed Modularity Index (MI) is discussed in Section 2.2.2. Their optimization method also uses a genetic algorithm (GA), which is presented in [28]. In their optimization routine, the modular decompositions are the individuals. Each decomposition is encoded in so called *extended genomes*. As a result of this, the GA itself is the clustering algorithm, without intervention of an algorithm such as LB-HMC.

The resulting decomposed DSM by Jung and Simpson is given in Figure 7. The results of the presented optimization routine of Section 6.1 are given in Figure 8.

In Figure 7, the modules seem rather dense and independent. However, the density and independence metrics used in LBDI nodes are much higher in Figure 8. A major contributor to this difference is the way the LBDI treats hierarchical relationships. The metrics of each system node that is recognized in the decomposition is expressed in terms of dependencies between its grandchildren and not its leaf nodes. A hierarchical decomposition requires the viewer to view each (sub-) system as a system-of-systems instead of viewing a single system-of-modules. Based on the LBDI value, this indicates that this DSM would benefit from more levels in its hierarchical decomposition.

The decomposition in Figure 8 is much more hierarchical. Its global bus is slightly different from the decomposition by Jung and Simpson. Most notably, the *Harness* has been moved to the third large sub-system. This may be due to its very asymmetric dependencies that negatively effect the LBDI's density metric, which causes it to remain a non-bus element during optimization.

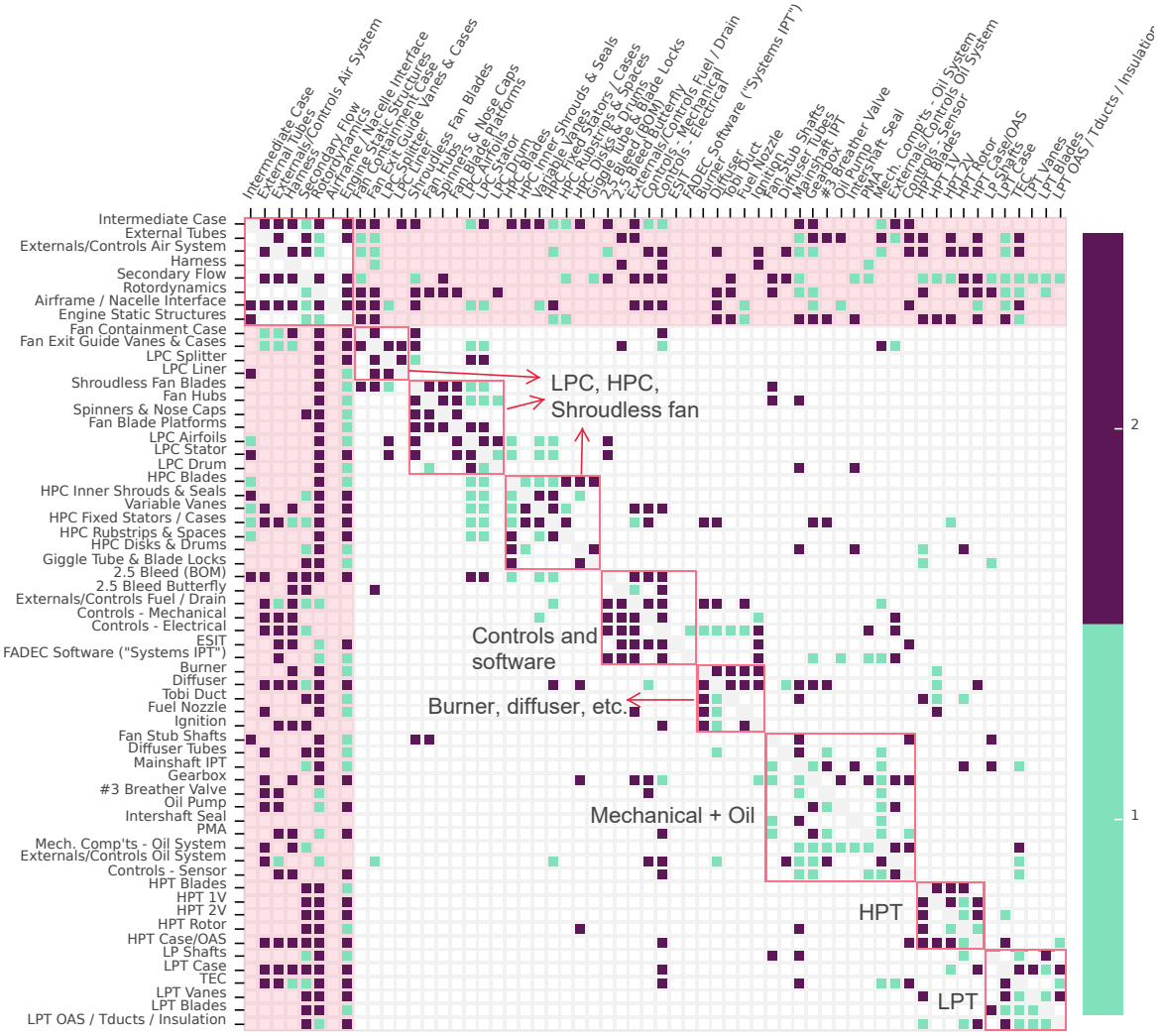


Figure 7: Decomposed DSM for the Pratt & Whitney Aircraft Engine as given by Jung and Simpson in [22]. Achieved LBDI($x_{\mathrm{w}} = 1.0, x_{\iota} = 0.25$) = 0.4823.

The *LPC*, *HPC* and *Shroudless fan* related components which are in separate modules in Figure 7 can be found in a more hierarchical decomposition in Figure 8. Even in Figure 7, the *LPC Airfoils* and *LPC Stator* can already be distinguished as the local bus module they form in Figure 8 because of their integrative function for all three modules. Deeper into the hierarchy of this module, the *Shroudless Fan Blades* and *Fan Exit Guide Vanes & Cases* are accurate local bus modules, too.

The controls and software module that follows in Figure 7 is not included in that capacity in Figure 8. In case of the latter, the software components have been more hierarchically spread out through the controls and actuation of the *Mechanical components* of the *Oil system, Diffuser, Fuel Nozzle, Ignition, Burner* and so forth. This may indicate that the single set of parameters for LB-HMC are not as adequate for this third sub-system as they are for the other sub-systems.

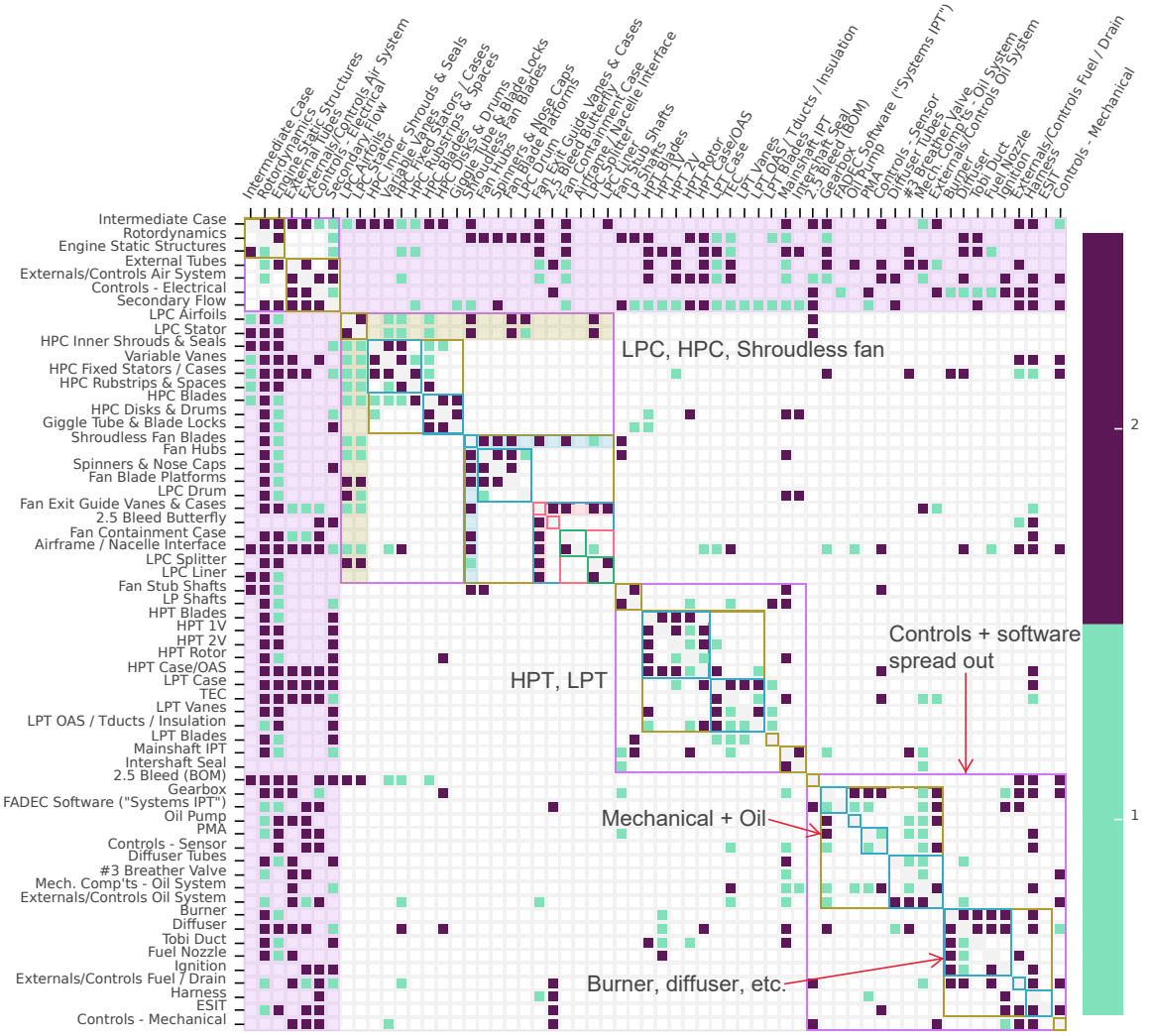


Figure 8: Decomposed DSM for the Pratt & Whitney Aircraft Engine. Achieved LBDI($x_{\mathrm{w}} = 1.0, x_{\iota} = 0.25$) = 0.7796 with LB-HMC parameters $\alpha = 2$, $\beta = 4.223$, $\mu = 2.385$ for HMC and $\gamma = 2.046$ for Gamma bus detection.

The *LPT* and *HPT* modules at the lower right in Figure 7 can be found in a hierarchical decomposition in the middle of Figure 8.

Following from these observations, the difference in the LBDI is best explained by the difference in hierarchical depth of both decompositions. The global outline of both decompositions is still similar, since most components end up in the same module or sub-system. Especially the third subsystem in the optimized decomposition seems to leave room for improvement. This may be due to the single set of parameters for HMC that is re-used for every sub-system when creating decompositions with the LB-HMC algorithm.

6.3 Complex Elevator System

The Complex Elevator System is a symmetrical DSM with multiple dependency types [24]. It describes a machine-room-less elevator called the 'Kone MonoSpace'. It is designed for low- to mid-rise buildings and uses permanent-magnet electric motors. The five defined dependency types are spatial, material, mechanical energy, electrical energy and information. It was published in a variation of 175 elements and a less granular variation of 45 elements. Granularity defines the 'grain size' or detail of the model of a system [29, 30]. In case of a DSM, this determines what its elements are considered to be. The less granular variation collapses a set of pre-defined modules and therefore contains less detail of the system.

The weights assigned to each dependency equal the number of active types for each dependency in the given DSM. Another way of interpreting this is that the weighted DSM is the sum of five binary DSMs – one for each dependency type. The number of active types in the given DSM varies between zero and three. The given optimized result in Figure 9 considers the fully granular DSM of 175 elements.

The complexity of the elevator becomes apparent in Figure 9. It shows two major sub-systems, ranging from the *Car guide rails* to the *Pit safety 1* and from the subsequent *Machinery 8* to the *Landing signalization*. From this decomposition, the numbered *Car components* seem to tie these major subsystems together. The *Car component 2*, *Car component 14* and *Electrical system 13* are in the global bus module, meaning their element degrees (number of related dependencies) are significantly higher than those of all other components.

The first major sub-system consists mostly of the components related to *Counterweights*, *Counterweight slings*, *Rope terminations*, *Pit components*, *Car floors*, *Car doors*, *Landing doors* and *Safety measures*. Therefore, this first major sub-system can be seen as the elevator car's supportive components and connections to the outside world, considering both the *Landing* and *Car* doors. The *Car guide rails* and *Counterweight screen* are detected as the bus modules, integrating these more supporting functions of the elevator. On a slightly deeper level, the *Counterweight guide rails* and *Bundle 2* are detected as the local bus modules. This seems a natural assignment, as guide rails, screens and bundles can interdepend with many components.

The second major sub-system consists mostly of the *Drive panels*, *Machinery*, *Control panel components*, *Interiors*, *Electrical systems* and *Signalization*. This combination can be seen as the user interaction with the elevator interacting with the machinery to drive the elevator. The dense block consisting of *Car components* and *Interior* stands out, revealing a strong dependency density. The second major sub-system is a very integrated system overall considering its high dependency density. The lack of bus modules emphasizes this, as none of the components have a high enough element degree for the given γ.

Further inspection of the results show that most high ranking decompositions with a high LBDI contain many hierarchical levels. Introducing another hierarchical level can have a major impact on the index value. An example thereof is the final element in the DSM, being the *Car electrification 2*. Since it has only a single dependency from and to the global bus module, it was not grouped with the other non-bus elements at the global level. This caused both previously discussed major sub-systems to end up in a separate grouping, essentially pushing them down a hierarchical level. Similar examples can be found in *Positioning System 1*, *Pit safety 1*, *Counterweight w/ hw* and *Installation accessories 1*. Considering the complexity of the Complex Elevator System, further studies could restrict the decomposition analysis to one dependency type at a time.

6.4 Discussion

Detecting decompositions with buses on a local level shifts from a system-of-modules towards a system-of-systems approach. The implementation of the LB-HMC algorithm uses the same set of parameters for all sub-systems for which it (re-) calculates the local decomposition. This results in sensible decompositions paired with the LBDI optimization, but it does not do justice to the system-of-systems approach completely. The parameters are optimized for all sub-systems at once, as opposed to finding locally optimized sub-systems.

The varying tree depth and number of children per node are challenges related to this that prove hard to overcome. The level of detail included in

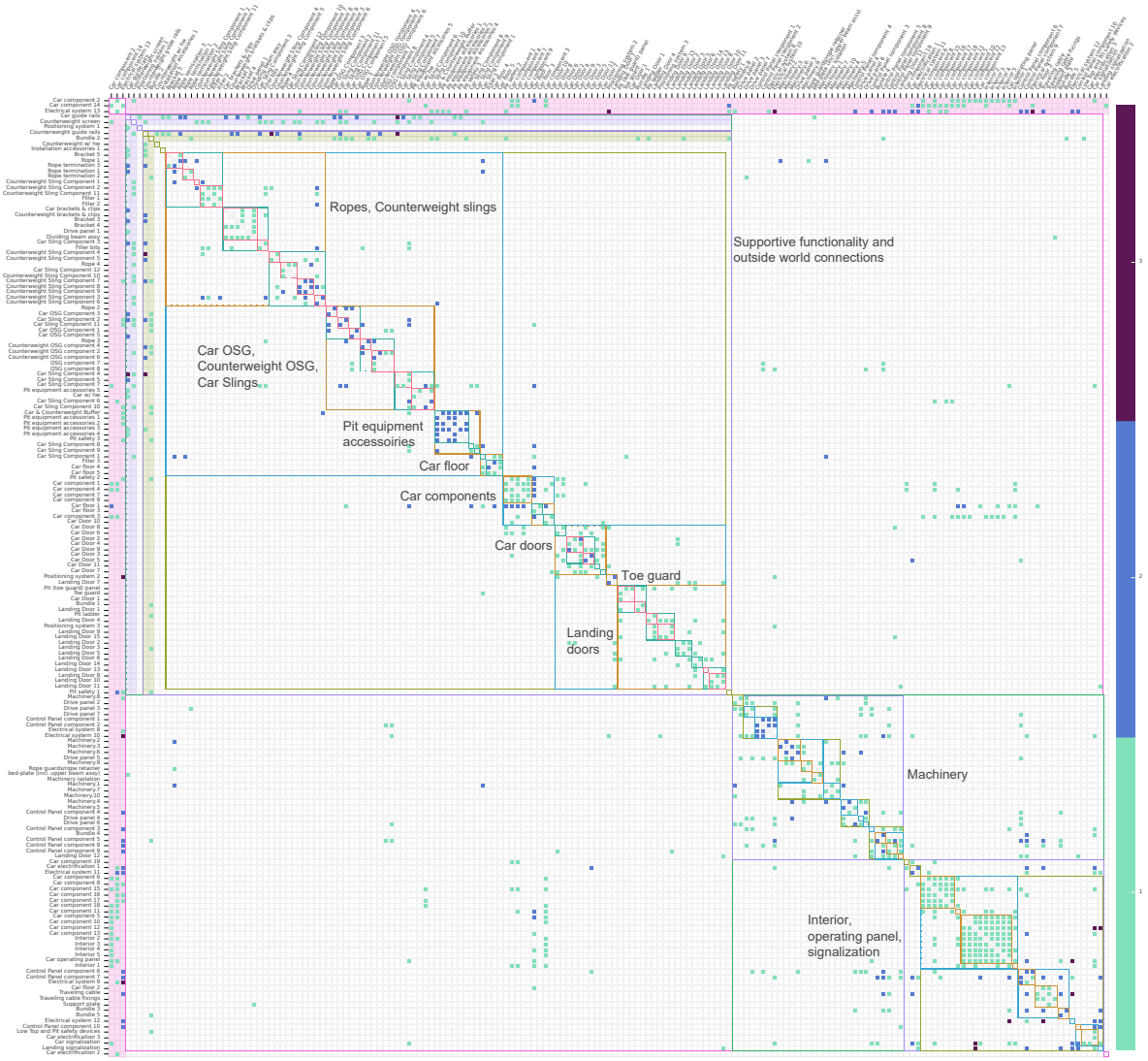


Figure 9: Decomposed DSM for the Complex Elevator System. Achieved LBDI($x_{\mathrm{w}} = 1.0, x_{\iota} = 0.25$) = 0.7796 with LB-HMC parameters $\alpha = 2$, $\beta = 4.387$, $\mu = 4.699$ for HMC and $\gamma = 4.080$ for Gamma bus detection.

the DSM is limited and its elements need not be of the exact same depth in the tree. When clustering bottom-up (agglomerative clustering such as HMC), the varying tree depth obfuscates what the actual influence on higher level systems will be. When clustering top-down (partitioning), it is not yet clear what the grandchildren of a system will be after subsequent partitioning. Both prevent local LBDI optimization during the agglomerative clustering or partitioning process.

Especially equalizing the size of modules and local decomposition refinements could yield better decompositions. Equalizing the size of modules can be rewarded using a metric similar to Model Description Length (MDL) as described in Section 2.2.1. The effort-based approach of MDL could also be used to steer the amount of hierarchical levels in a decomposition. For implementation in the current index, that would require these to be defined as normalized metrics.

7 Conclusion

This work presents a new algorithm that calculates hierarchical decompositions with local bus modules for a DSM. Additionally, a novel index is presented that can be used to rank different decompositions for a DSM by their decomposition quality.

The algorithm relies on both a hierarchical clustering algorithm and a bus detection algorithm. It creates an initial hierarchical decomposition that is subjected to a top-down bus detection routine. If bus elements are found for some sub-system in the decomposition, both the bus elements and remaining non-bus elements are re-clustered. The results of this re-clustering are inserted into the initial decomposition such that it contains a bus module on a local level. The algorithm can be implemented using any hierarchical clustering algorithm and any bus detection algorithm. The given implementation uses Hierarchical Markov Clustering and Gamma bus detection and is called LB-HMC.

The proposed Local Bus Decomposition Index (LBDI) requires a DSM decomposition as a rooted, directed, labeled tree and a DSM's adjacency matrix as input. Multiple sub-systems can be recognized in a decomposition. The index value consists of a module independence metric and a module density metric that is calculated for each subsystem. Sub-systems with more elements contribute more to the final index value.

A Simple Genetic Algorithm (SGA) is used to tune the algorithm parameters of LB-HMC for both the Pratt & Whitney Aircraft Engine and Complex Elevator System case studies to maximize the LBDI. The obtained hierarchical decompositions with local buses show significant improvements in terms of the new decomposition index when compared to the results for the Pratt & Whitney Aircraft Engine in [22]. The Complex Elevator System's results show that the SGA yields insightful results for larger matrices, as this DSM features 175 elements.

The LB-HMC algorithm shows promising performance, despite the a addition of a local bus search routine. The SGA implementation in Python 3 calculated roughly 3000 decompositions for this DSM in about 5 minutes time on a system with an Intel Core i5-6300HQ CPU and 8 GB of RAM.

8 Future work

The proposed algorithm currently yields hierarchical decompositions with buses on a local level. This shifts the paradigm from a system-of-modules towards a system-of-systems approach. The implementation of the algorithm uses identical parameters for all sub-systems it (re-) calculates the local decomposition for. This does not do justice to the system-of-systems approach completely. Local optimization of parameters may result in decompositions of higher quality.

The proposed index is sensitive to the introduction of more hierarchical levels in any sub-system. Sometimes, the addition of a single element to a sub-system drastically changes the index value because of the change it induces in the hierarchy of the decomposition. This effect may be reduced by implementing a metric similar to the Model Description Length (MDL) metric [13]. An effort-based approach similar to MDL could help to equalize the size of modules in each sub-system and the number of levels in the hierarchical decomposition. As such, it could result in a more robust index value.

References

[1] Paul Rook. "Controlling software projects". In: *Softw. Eng. J.* 1.1 (1986), pp. 7–16. DOI: 10.1049/sej:19860003.

[2] Steven D. Eppinger and Tyson R. Browning. *Design Structure Matrix - Methods and Applications*. 2012.

[3] Tyson R. Browning. "Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities". In: *IEEE Trans. Eng. Manag.* 63.1 (2016), pp. 27–52. DOI: 10.1109/TEM.2015.2491283.

[4] Giota Paparistodimou et al. "System Architectures Assessment Based On Network Metrics". In: *19th Int. Depend. Struct. Model. Conf. DSM 2017*. 2017, pp. 117–126.

[5] Thomas U Pimmler and Steven D Eppinger. "Integration Analysis of Product Decompositions". In: *ASME Des. Theory Methodol. Conf.* Minneapolis, 1994.

[6] T. Wilschut et al. “Multilevel Flow-Based Markov Clustering for Design Structure Matrices”. In: *J. Mech. Des.* 139.12 (2017), p. 121402. DOI: 10.1115/1.4037626.

[7] Patrik Eklund et al. “The Logic of DSM”. In: *19th Int. Depend. Struct. Model. Conf. DSM 2017*. 2017, pp. 25–32.

[8] Andrew Harold Tilstra, Carolyn Conner Seepersad, and Kristin L. Wood. “A high-definition design structure matrix (HDDSM) for the quantitative assessment of product architecture”. In: *J. Eng. Des.* 23.10-11 (2012), pp. 767–789. DOI: 10.1080/09544828.2012.706748.

[9] Manuel E. Sosa, Steven D. Eppinger, and Craig M. Rowles. “Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions”. In: *J. Mech. Des.* 125.2 (2003), p. 240. DOI: 10.1115/1.1564074.

[10] David M. Sharman and Ali A. Yassine. “Characterizing complex product architectures”. In: *Syst. Eng.* 7.1 (2004), pp. 35–60. DOI: 10.1002/sys.10056.

[11] J MacQueen. “Some methods for classification and analysis of multivariate observations”. In: *Proc. Fifth Berkeley Symp. Math. Stat. Probab.* 1967. DOI: citeulike-article-id:6083430.

[12] Somwrita Sarkar et al. “Spectral Characterization of Hierarchical Modularity in Product Architectures 1”. In: *J. Mech. Des.* 136.1 (2013), p. 011006. DOI: 10.1115/1.4025490.

[13] Tian-Li Yu, Ali A. Yassine, and David E. Goldberg. “An information theoretic method for developing modular architectures using genetic algorithms”. In: *Res. Eng. Des.* 18.2 (2007), pp. 91–109. DOI: 10.1007/s00163-007-0030-1.

[14] Stijn Van Dongen. “Graph Clustering Via a Discrete Uncoupling Process”. In: *SIAM J. Matrix Anal. Appl.* 30.1 (2008), pp. 121–141. DOI: 10.1137/040608635.

[15] Fionn Murtagh and Pedro Contreras. “Algorithms for hierarchical clustering: An overview”. In: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 2.1 (2012), pp. 86–97. DOI: 10.1002/widm.53. arXiv: 1105.0121.

[16] Fionn Murtagh and Pedro Contreras. “Algorithms for hierarchical clustering: an overview, II”. In: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 7.6 (2017), e1219. DOI: 10.1002/widm.1219.

[17] Rui Xu and D. WunschII. “Survey of Clustering Algorithms”. In: *IEEE Trans. Neural Networks* 16.3 (2005), pp. 645–678. DOI: 10.1109/TNN.2005.845141.

[18] T. Wilschut et al. “Multi-Level Flow-Based Markov Clustering For Design Structure Matrices”. In: *Int. Des. Eng. Tech. Conf.* (2016), pp. 1–10.

[19] Steven H. Strogatz. “Exploring complex networks”. In: *Nature* 410.6825 (2001), pp. 268–276. DOI: 10.1038/35065725. arXiv: 0102091 [cond-mat].

[20] Manuel Sosa, Jurgen Mihm, and Tyson Browning. “Degree Distribution and Quality in Complex Engineered Systems”. In: *J. Mech. Des.* 133.10 (2011), p. 101008. DOI: 10.1115/1.4004973.

[21] Katja Hölttä-Otto et al. “Comparative analysis of coupling modularity metrics”. In: *J. Eng. Des.* 23.10-11 (2012), pp. 790–806. DOI: 10.1080/09544828.2012.701728.

[22] Sangjin Jung and Timothy W. Simpson. “New modularity indices for modularity assessment and clustering of product architecture”. In: *J. Eng. Des.* 28.1 (2017), pp. 1–22. DOI: 10.1080/09544828.2016.1252835.

[23] Katja Hölttä-Otto and Olivier de Weck. “Degree of Modularity in Engineering Systems and Products with Technical and Business Constraints”. In: *Concurr. Eng.* 15.2 (2007), pp. 113–126. DOI: 10.1177/1063293X07078931.

[24] Valtteri Niutanen et al. “Complex Elevator System DSM - Case for a DSM Design Sprint”. In: *19th Int. Depend. Struct. Model. Conf. DSM 2017*. 2017.

[25] Craig M. Rowles. “System integration analysis of a large commercial aircraft engine”. In: (1999).

[26] Thomas Back, D.B Fogel, and Z Michalewicz. “Evolutionary Computation 1: Basic Algorithms and Operators”. In: *Evol. Comput.* (2000), p. 378. DOI: 10.1016/B978-044452701-1.09002-5.

[27] Manuel E Sosa, Steven D Eppinger, and Craig M Rowles. “The Misalignment of Product Architecture and Organizational Structure in Complex Product Development”. In: *Manage. Sci.* 50.12 (2004), pp. 1674–1689. DOI: 10.1287/mnsc.1040.0289.

[28] S. Jung and T.W. Simpson. “A clustering method using new modularity indices and a genetic algorithm: With extended Chromosomes”. In: *J. Mod. Proj. Manag.* 3.2 (2015), pp. 38–45.

[29] Noemi Chiriac et al. “Level of Modularity and Different Levels of System Granularity”. In: *J. Mech. Des.* 133.10 (2011), p. 101007. DOI: 10.1115/1.4005069.

[30] Jakob F Maier, Claudia M Eckert, and P. John Clarkson. “Model granularity in engineering design concepts and framework”. In: *Des. Sci.* 3 (2017), e1. DOI: 10.1017/dsj.2016.16.