Euro. Jnl of Applied Mathematics (2019), *vol.* 30, *pp.* 278–297.
doi:10.1017/S0956792518000086

Physically feasible decomposition of Engino® toy models: A graph-theoretic approach†

E. N. ANTONIOU[1], A. ARAÚJO[2], M. D. BUSTAMANTE[3] and A. GIBALI[4]

[1]*Department of Information Technology, Alexander Technological Educational Institute of Thessaloniki, 57400, Thessaloniki, Greece*
email:antoniou@it.teithe.gr
[2]*CMUC, Department of Mathematics, University of Coimbra, Coimbra, Portugal*
email: alma@mat.uc.pt
[3]*Institute for Discovery, School of Mathematics and Statistics, University College Dublin, Belfield, Dublin 4, Ireland*
email: miguel.bustamante@ucd.ie
[4]*Department of Mathematics, ORT Braude College, P.O. Box 78, Karmiel 2161002, Israel*
email: avivg@braude.ac.il

(*Received 27 July 2017; revised 29 January 2018; accepted 29 January 2018; first published online 4 March 2018*)

During the 125th European Study Group with Industry held in Limassol, Cyprus, 5–9 December 2016, one of the participating companies, Engino.net Ltd, posed a very interesting challenge to the members of the study group. Engino.net Ltd is a Cypriot company, founded in 2004, that produces a series of toy sets – the Engino® toy sets – consisting of a number of building blocks, which can be assembled by pupils to compose toy models. Depending on the contents of a particular toy set, the company has developed a number of models that can be built utilizing the blocks present in the set; however, the production of a step-by-step assembly manual for each model could only be done manually. The goal of the challenge posed by the company was to implement a procedure to automatically generate the assembly instructions for a given toy. In the present paper, we propose a graph-theoretic approach to model the problem and provide a series of results to solve it by employing modified versions of well-established algorithms in graph theory. An algorithmic procedure to obtain a hierarchical, physically feasible decomposition of a given toy model, from which a series of step-by-step assembly instructions can be recovered, is proposed.

Key words: 05C38, 05C90, 68R10, 94C15.

1 Introduction

Engino® toy models are created by assembling small blocks or bricks together, with the purpose of helping pupils build technological models creatively and easily so that they can experiment and learn about science and technology in a playful way. Each of the toy sets produced by Engino.net Ltd has a specific number of blocks that can be assembled into many different models. It has been observed that the creative potential of each toy

† MDB acknowledges support from Science Foundation Ireland under research Grant number 12/IP/1491.

set increases exponentially as the number of blocks in the set increases. This is due to the patented design of the Engino® blocks that allow connectivity on many directions in three-dimensional (3D) space simultaneously.

To demonstrate the creative potential of its toy sets, the company has developed a large number of toy models that can be built using the contents of the set. The ingredients and the connections required to obtain each particular toy model have been recorded in a database system. Despite the detailed recording, the production of step-by-step instructions for the assembly of a particular toy model has been proved to be a tedious task that has to be accomplished manually. This is mainly due to the 3D nature of the models and the complexity of the interconnections between the blocks, which in many cases impose a particular order in the steps that have to be taken to assembly the structure. The goal of the challenge posed by the company during the 125th European Study Group with Industry was the development of an automatic procedure able to produce step-by-step assembly instructions manual for every toy model that has been recorded in the company's databases.

To accomplish this task, we propose a graph-theoretic approach. Given a toy model, we associate with it a directed graph whose vertices correspond to the building blocks of the model and whose edges represent physical connections between the two blocks (see [11] and references therein). Moreover, in order to partially capture the actual geometry of the toy model, every edge of the graph is labelled with a vector showing the direction of the underlying physical connection in 3D space. This labelling of the edges provides an adequate description of the geometry of the model, for the purposes of our application. It should be however noted, that the exact geometry of the (possibly multiple) connections between individual blocks of a particular model has been recorded in full detail and this information is available at the final stage of the assembly instructions generation.

With this setup, in order to produce the assembly instructions of a given model, we first apply the reverse process recursively. Given a description of a toy model, and hence its associated graph, we develop a method to break it apart into clusters of blocks in a manner that is physically possible. In what follows, we call this procedure a *physically feasible decomposition (PFD)* of the model. The result of such a decomposition is a collection of sub-models or components on which the method can be recursively applied until no further decompositions are possible. Thus, a characterization of PFD of a model is of fundamental importance in the decomposition procedure. The outcome of this separation process is a hierarchical tree structure of components, whose nodes can traversed in a postorder fashion, to generate an ordered sequence of nodes, which in turn dictate a series of step-by-step assembly instructions.

The problem of determining a series of steps required to decompose a complex structure into its constituent components has been the subject of several studies dating back to the 1980s. This class of problems is termed *disassembly sequencing* and depending on the nature of the underlying structure, a number of different approaches has been employed (see [8] for an extensive survey). The motivation behind the study of disassembly sequencing originates mainly from the fact that by reversing the steps of a disassembly sequence, one can obtain an assembly procedure of the structure under the study. In this respect, disassembly sequences are closely related to the automated generation of assembly instructions of complex structures (see, for instance [1, 6, 9]).

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

The procedure proposed in the present paper can be compared to the one presented in [9] for the computation of a hierarchical explosion graph. Contrary to the approach used in [1,9] for the construction of the explosion graph, which detaches individual parts one-by-one from the structure, and in turn applies a search strategy for the extraction of the hierarchy of components, our method produces directly a physically feasible decomposition (PFD) into components along a given spatial direction. As shown in Section 4, a maximal PFD can be obtained using well-known linear-time algorithms and the recursive application of this procedure results in a hierarchical decomposition, which is comparable to the hierarchical explosion graph in [9].

The contents of the paper are organized as follows: In Section 2, we briefly recall some basic concepts and facts from graph theory required for the development of our results in the sequel. In the subsequent section, we present the proposed graph-theoretic framework and through a series of motivating examples we introduce the notion of a *PFD* of a toy model. In the same section, we also define the *component connectivity graph (CCG)* implied by the removal of a set of edges of the model's graph and show that such a removal gives rise to a PFD if and only if the corresponding CCG is a directed acyclic graph (DAG). In Section 4, we define maximal PFDs along a given direction and show that such decompositions can be obtained by applying well established, linear-time algorithms used for the discovery of strongly connected components in directed graphs. In Section 5, we outline an algorithmic procedure to obtain a hierarchical decomposition of a given toy model, using as intermediate steps for such a decomposition, maximal PFDs along appropriately chosen spatial directions. Moreover, at the end of Section 5, the resulting hierarchical decomposition of the model is utilized to recover a series of assembly instructions. Finally, in Section 6, we review and summarize our results.

2 Graph theory prerequisites

In this section, we review a number of definitions and facts from graph theory that will be instrumental in the sequel. Most of these definitions and results can be found in [2,3].

A *directed graph* G, denoted by $G(V,E)$, is an ordered pair of sets (V,E), where

- V is the set *vertices* or nodes of G;
- E is the set of *directed edges* consisting of directed pairs (u,v), where $u,v \in V$.

Moreover, if E is allowed to be a multiset instead of a set, then $G(V,E)$ is a *directed multigraph*. On the other hand, if pairs of the form (v,v), (called *loops*) are not allowed in E, then $G(V,E)$ is a *directed simple graph*. Similar definitions can be given in case the edge set (multiset) has elements undirected pairs of vertices. In such a case, the (multi)graph is called *undirected*.

A graph $G_1(V_1,E_1)$ is a *subgraph* of a given graph $G(V,E)$ if $V_1 \subseteq V$ and $E_1 \subseteq E$ consist exclusively of edges having both its endpoints in V_1. Moreover, for $V_1 \subseteq V$, we define the *induced subgraph* $G[V_1]$ as the subgraph of $G(V,E)$, whose vertex set is V_1 and its edge set is the set of all edges of E, having both their endpoints in V_1.

In a directed graph $G(V,E)$, a *directed (resp. undirected) path* of length k, starting from v_0 and ending to v_k, is a sequence of vertices $v_0,v_1,\ldots,v_k$, such that $(v_i,v_{i+1}) \in E$ (*resp.* $(v_i,v_{i+1}) \in E$ or $(v_{i+1},v_i) \in E$), for all $0 \leqslant i < k$. In case $v_0 = v_k$ and $k > 0$ the path

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

is called a *directed (resp. undirected) cycle*. A vertex $t \in V$ is said to be *reachable* from $s \in V$, if there exists a directed path from s and to t.

A directed graph $G(V, E)$ is set to be a *DAG*, if it contains no directed cycles, or equivalently, if there exists no vertex in V, which is non-trivially reachable from itself. A *topological ordering* of the vertices of a directed graph $G(V, E)$ is a total ordering of its vertices $v_1, v_2, \ldots, v_n$, such that for all $(v_i, v_j) \in E$, $i \leqslant j$ holds.

Theorem 2.1 *A directed graph $G(V, E)$ is acyclic if and only if a topological ordering of its vertices exists.*

A directed graph $G(V, E)$ is called *strongly (resp. weakly) connected* if for every pair of vertices $u \in V$, $v \in V$, there exists a directed (resp. undirected) path from u to v. A maximal strongly (resp. weakly) connected subgraph of a graph, i.e., a strongly connected subgraph, which is not a proper subgraph of any other strongly connected subgraph, is called a *strongly (resp. weakly) connected component*.

The *condensation* of a directed graph $G(V, E)$ is a directed graph $G_{co}(V_{co}, E_{co})$, with

- $V_{co} = \{C_i : C_i \text{ is a strongly connected component of } G(V, E)\}$;
- $E_{co} = \{(C_i, C_j) : \exists (u, v) \in E \text{ such that } u \in C_i, v \in C_j\}$.

Theorem 2.2 *The condensation of any directed graph $G(V, E)$ is a DAG.*

A *tree* is an undirected graph in which every pair of vertices is connected *via* a unique path. A *rooted tree* is a tree having one particular vertex designated as its *root node*. An *ordered tree* is a rooted tree in which an ordering is specified for the children of each vertex. A *binary tree* is a rooted tree in which every vertex has at most two children. A binary tree is *full* if every node has either zero or two children.

3 Physically feasible decomposition of toy models

We now present the proposed framework for the solution of the decomposition problem discussed above based on a graph-theoretic approach. Given a toy model $\mathscr{M}$, we associate to it a directed graph $G(V, E)$, where

- $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set of G with each vertex v_i corresponding to a block of $\mathscr{M}$;
- $E = \{(u, v) : u, v \in V\}$ is the edge set of G with each directed edge representing a connection between two blocks of the model.

Every physical connection between two blocks of the model can be aligned in space to one particular direction vector, chosen out of a finite collection of directions. For instance, if a model uses only perpendicular connections between its blocks in 3D space, we can identify three direction vectors $\hat{i}, \hat{j}, \hat{k}$ along which all connections can be aligned. A connection between two blocks of the model u, v, aligned to a particular direction $\hat{d}$ in physical space, gives rise to a directed edge $(u, v) \in E$, if the vector from u to v points towards the same direction as $\hat{d}$.

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

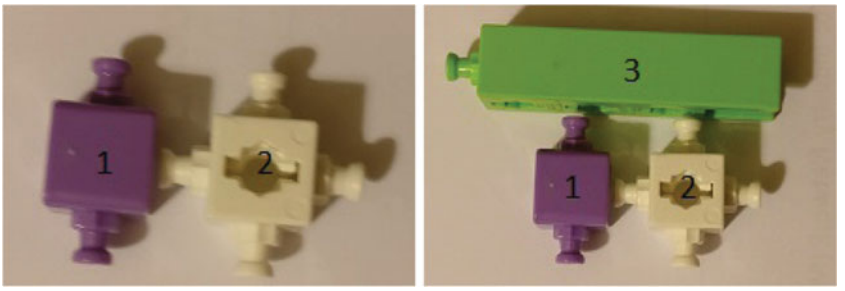

FIGURE 1. Two blocks that can be disconnected (left); blocks 1, 2 cannot be disconnected (right).

Assuming that all the connections of the model $\mathscr{M}$ correspond to p, not necessarily orthogonal, distinct spatial directions $\hat{d}_i$, we can partition the edge set E into a family of p mutually disjoint sets E_i, $i = 1, 2, \ldots, p$, each of which contains the edges associated to connections sharing the same direction in space. It should be noted that the physical connections between the blocks of the toy model $\mathscr{M}$, are assumed to be fixed, meaning that the resulting construction is rigid and contains no moving or rotating parts. Thus, the only possible way to separate two connected blocks is to apply opposite forces along the physical direction $\hat{d}_i$ associated to the connection, provided that resulting the displacement is physically feasible in the sense described in the paragraph that follows. At first this may seem to be a rather restrictive assumption with respect to the types of toy models it allows to be constructed, as there are many actual toy compositions in Engino's collection involving moving or rotating parts. However, as discussed with representatives of the company during the 125th ESGI meeting, in most such cases the moving or rotating parts can either be considered as separate rigid submodels (e.g., a two wheel and axle submodel), or their connection to the rest of the model is non-fixed (e.g., a pinned joint), allowing them to be detached from it by pulling them along some non-blocking direction. In view of this setup, we propose the following principle to describe the conditions under which a disconnection of two blocks is physically possible.

Physically feasible disconnection of two blocks: In order to disconnect two blocks corresponding to vertices $v_1, v_2 \in V$, connected *via* an edge $(v_1, v_2) \in E$ aligned to a given spatial direction $\hat{d}_i$, the blocks v_1, v_2 must be able to be displaced along the directions $-\hat{d}_i, \hat{d}_i$, respectively, when appropriate opposite forces are applied on the blocks.

The idea behind the above principle is illustrated in the following example.

Example 3.1 Consider the blocks shown in Figure 1. In the left side, the blocks 1, 2 can be disconnected using two opposite horizontal forces, since their application on the two blocks will result in displacements along the horizontal direction. If a third block is added as shown in right side of Figure 1, then the blocks 1, 2 cannot be disconnected by applying on them opposite horizontal forces, since their displacement is blocked by their vertical connections to the block number 3.

The idea of disconnecting two blocks of the model in a physically feasible manner can be easily generalized to describe the corresponding decomposition of a model into two submodels. In general, the removal of a set of edges along a given direction may

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

result into a decomposition of the graph of the model into two or more weakly connected components. However, not all such removals can be actually applied on the physical model to decompose it into two or more submodels. This is due to the fact that in certain cases the physical displacement of the resulting weakly connected components of the model is blocked by other physical connections, due to the presence of edges not removed in the current phase.

We can extend the principle of physically feasible disconnection, introduced above, to the case of the separation of two weakly connected components.

Physically feasible decomposition into two components: The removal of a set of edges, aligned to a particular space direction $\hat{d}_i$, is physically feasible, if and only if the two resulting weakly connected components are able to be displaced along the directions $-\hat{d}_i, \hat{d}_i$, respectively, when appropriate opposite forces are applied on these blocks.

For brevity, in what follows, we shall call this decomposition a *2-PFD of the model*. The above decomposition is equivalent to assuming that, during the separation process, each of the two weakly connected components behaves like a single block, but unlike the single blocks case, it is possible to have multiple parallel connections between them.

Our next goal is to obtain a characterization of 2-PFD's that are possible along a given direction. In this respect, it is instrumental to introduce the notion of the CCG of a model $\mathscr{M}$, implied by the removal of a set of co-linear edges, which provides a higher level view of the decomposition.

Definition 3.2 (Component Connectivity Graph) Let $G(V, E)$ be the graph associated to a model $\mathscr{M}$ and $\bar{E}_i \subseteq E_i$ a non-empty set of edges, where E_i is the set of all edges of $G(V, E)$ along the spatial direction $\hat{d}_i$. The CCG, implied by the removal of the edges of $\bar{E}_i$, is a directed graph, $G_C(V_C, E_C)$, whose vertices are the weakly connected components C_i, $i = 1, 2, \ldots, k$, into which $G(V, E)$ is partitioned with the removal of the edges of $\bar{E}_i$. Two components C_i, C_j are connected *via* an edge $(C_i, C_j) \in E_C$ if and only if $i \neq j$, and there exists an edge $(v, u) \in \bar{E}_i$ with $v \in C_i$ and $u \in C_j$.

We should note that according to the above definition the CCG implied by the removal of a set of edges $\bar{E}_i \subseteq E_i$ is a simple directed graph, since by construction it cannot contain neither loops nor parallel edges sharing the same source and target vertices. The above ideas are illustrated in the following example.

Example 3.3 Consider the model shown in Figure 2. The graph $G(V, E)$ of the model is depicted in Figure 3, where $\hat{d}_1, \hat{d}_2$ are respectively the horizontal (left–right) and vertical (bottom–up) direction vectors.

If we remove all edges along the horizontal direction, i.e., edges $(2, 3)$, $(1, 5)$ and $(4, 5)$, the graph is decomposed into two weakly connected components $C_1 = \{1, 2, 3, 4\}$ and $C_2 = \{5\}$ as shown in Figure 4, and the implied CCG by this removal of edges is shown in Figure 5. Clearly, nothing prevents the displacement of the two components C_1, C_2 from moving towards $-\hat{d}_1, \hat{d}_1$, respectively, when appropriate horizontal forces are applied on them. Thus, the removal of all horizontal edges implies a 2-PFD of the model.

On the other hand, if we choose to remove all edges along $\hat{d}_2$, we end up with the weakly connected components C_1', C_2' shown in Figure 6, and the corresponding CCG is

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

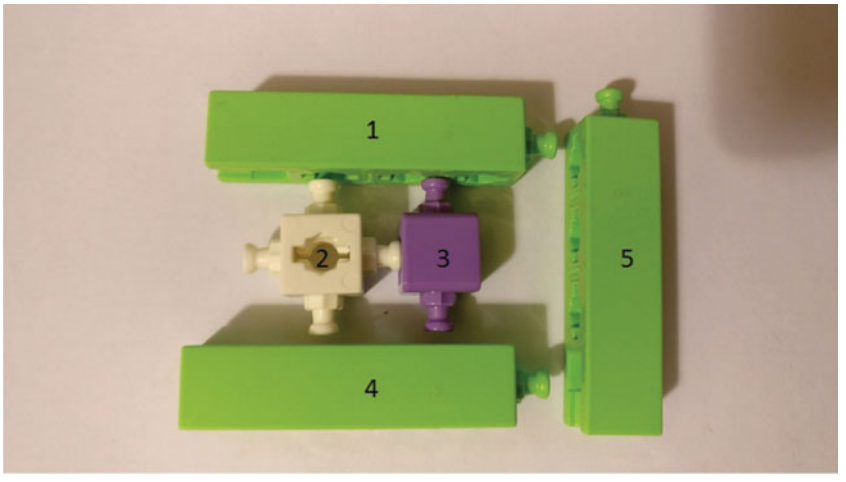

FIGURE 2. A picture of the actual model.

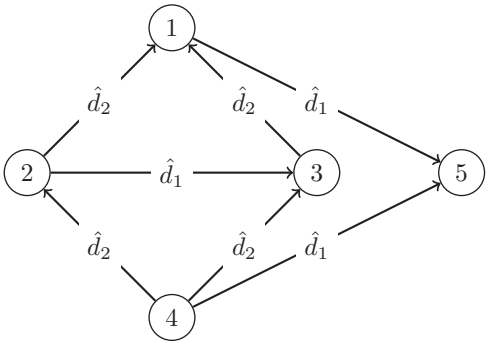

FIGURE 3. The graph $G(V, E)$ of the model.

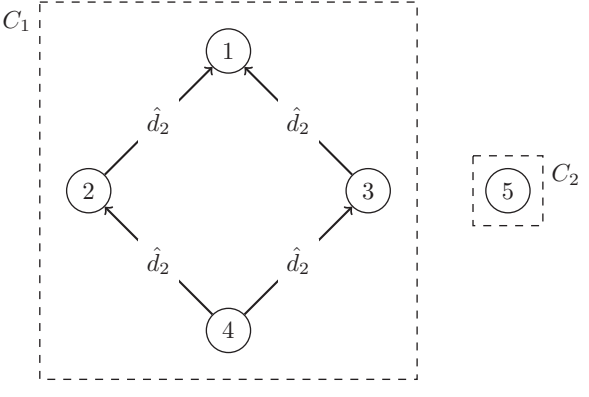

FIGURE 4. The graph $G(V, E)$ after the removal of all edges along $\hat{d}_1$.

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

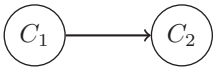

FIGURE 5. The CCG of the model after the removal of all edges along $\hat{d}_1$.

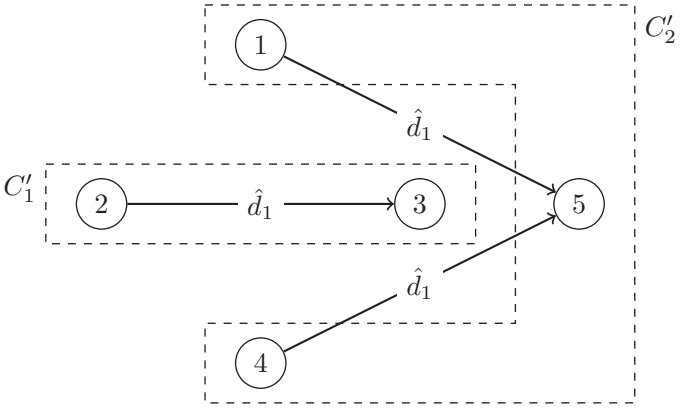

FIGURE 6. The graph $G(V, E)$ after the removal of all edges along $\hat{d}_2$.

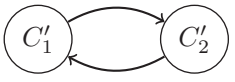

FIGURE 7. The CCG of the model after the removal of all edges along $\hat{d}_2$.

the one in Figure 7. Despite the fact that the removal of the four vertical edges separates the graph into two weakly connected components, it is clear that such a decomposition is not physically feasible. Obviously, the blocks $2, 3$ of C_1' cannot be displaced vertically, because they are 'trapped' between the components $1, 4$ of C_2'.

In view of the decomposition along the spatial direction $\hat{d}_1$ shown in Example 3.3, it becomes apparent that not all the edges removed correspond to a physically feasible disconnection of two blocks. This is the case with the edge $(2, 3)$ in the graph of Example 3.3, which does not appear in Figure 4 due to its removal. Despite the fact that this edge can be theoretically removed during the removal of all edges along $\hat{d}_1$, the blocks $2, 3$ cannot be disconnected because the perpendicular connections with blocks $1, 4$ obstruct their horizontal displacement. On the other hand, the edges $(1, 5)$, $(4, 5)$ obviously contribute actively on the decomposition of the graph into two components C_1 and C_2, shown in Figure 8. The distinguishing property between these two types of edges is that the former has both its endpoints on the same weakly connected component after the removal of all edges along $\hat{d}_1$, while each of the latter type of edges have their start and end points lying on distinct components. The edges that actively contribute to the formation of weakly connected components of a given CCG, will be called *physically removable* for the given CCG. A maximal subset of physically removable edges, along a given spatial direction, can be successfully computed using the technique presented in Section 4.

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

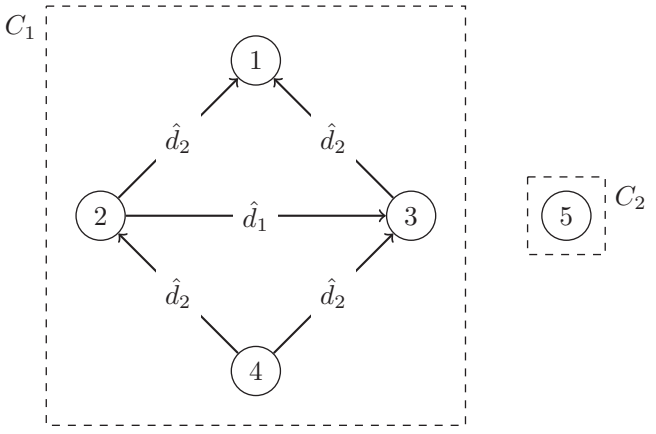

FIGURE 8. The graph $G(V, E)$ after the removal of all edges along $\hat{d}_1$, which are physically removable.

Proceeding a step further, we can provide a characterization of 2-PFD's in terms of a particular property of the edges connecting the weakly connected components in the corresponding CCG.

Lemma 3.4 *Let $\mathscr{M}$ be a toy model with the associated directed graph $G(V, E)$. Assume that the removal of a non-empty set of edges $\bar{E}_i \subseteq E_i$, where E_i is the set of all edges of $G(V, E)$ along the direction $\hat{d}_i$, gives rise to the CCG, $G_C(V_C, E_C)$, where $V_C = \{C_1, C_2\}$. Then, the removal of the edges of $\bar{E}_i$ is a* 2-PFD of $\mathscr{M}$ if and only if E_C contains exactly one of the edges (C_1, C_2), (C_2, C_1).

Proof We first note that since $\bar{E}_i$ is non-empty, so is E_C. Moreover, recall that $G_C(V_C, E_C)$ is simple, so, E_C will either contain exactly one or both (C_1, C_2), (C_2, C_1). Assume now that E_C contains both (C_1, C_2) and (C_2, C_1). Then, due to the presence of (C_1, C_2), in order to separate C_1 from C_2 we should be able to displace C_1 towards $-\hat{d}_i$ and C_2 towards $\hat{d}_i$, by applying appropriate opposite forces on C_1 and C_2. On the other hand, due to the presence of (C_2, C_1), in order to accomplish the same task, C_1 should be able to move towards $\hat{d}_i$ and C_2 towards $-\hat{d}_i$, using again appropriate opposite forces. Obviously, neither C_1 nor C_2 can move simultaneously on both spatial directions $-\hat{d}_i$, $\hat{d}_i$. Thus, the removal of the edges of $\bar{E}_i$, is not a 2-PFD, which proves the 'only if' part of the lemma.

Conversely, assume without loss of generality that E_C contains only (C_1, C_2). This means that, in physical space, the components C_1,C_2 are connected only on one side, leaving their externally exposed sides free (see Figure 9). Thus, removing the edges of $\bar{E}_i$ connecting the vertices of C_1 to those of C_2, will result in a 2-PFD of the model, since C_1 can be displaced towards the direction of $-\hat{d}_i$ and C_2 towards that of $\hat{d}_i$. □

Proceeding a step further, we can generalize the idea of a PFD into the case where the removal of a set of edges, along a particular spatial direction $\hat{d}_i$, separates the model into more than two weakly connected components. Assume that after the removal of a set of edges $\bar{E}_i \subseteq E_i$, we end up with $k > 2$ components. Such a decomposition is physically

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

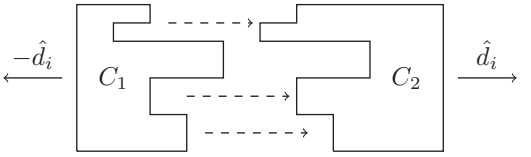

FIGURE 9. 2-PFD of C_1, C_2.

feasible if we can obtain it by applying a 2-PFD of the original model by removing an appropriate subset of edges of $\bar{E}_i$, and in turn by repeating 2-PFD procedures on the resulting submodels, recursively. A PFD giving rise to $k > 2$ components, that can be accomplished recursively by applying a series 2-PFD's, will be called a k-PFD.

The above idea is formalized in the following definition.

Definition 3.5 (k-PFD) Let $\mathscr{M}$ be a toy model and $G(V, E)$ its associated directed graph. Assume that the removal of a non-empty set of edges $\bar{E}_i \subseteq E_i$, where E_i is the set of all edges of $G(V, E)$ along the direction $\hat{d}_i$, gives rise to the CCG, $G_C(V_C, E_C)$, consisting of $k \geqslant 2$ weakly connected components. We say that the removal of the edges $\bar{E}_i$ implies a k-PFD of the model $\mathscr{M}$, if there exists a set of edges $\bar{E}_i^0 \subseteq \bar{E}_i$, whose removal implies a 2-PFD of $\mathscr{M}$ into C_1, C_2, for which exactly one of the following is true:

- $C_1 \in V_C$ and $C_2 \in V_C$;
- $C_1 \in V_C$ and the removal of all edges of $\bar{E}_i \backslash \bar{E}_i^0$ from C_2, implies its $(k-1)$-PFD;
- $C_2 \in V_C$ and the removal of all edges of $\bar{E}_i \backslash \bar{E}_i^0$ from C_1, implies its $(k-1)$-PFD;
- $C_j \notin V_C$, for $j = 1, 2$ and appropriate removal of edges of $\bar{E}_i \backslash \bar{E}_i^0$ from each one of them, implies a k_1-PFD of C_1 and a k_2-PFD of C_2, such that $k_1 + k_2 = k$.

If the removal of any set of edges $\bar{E}_i \subseteq E_i$, results in a CCG with only one weakly connected component, we say that we have a *1-PFD* or a *non-PFD* of the model.

Remark 3.6 The structure of a k-PFD of a model $\mathscr{M}$ can be represented by a full, ordered, binary tree T, having as its root node the entire vertex set V_C. The internal nodes of T are subsets of V_C corresponding to weakly connected components of G_C resulting in each step of the recursive application of 2-PFD's. Finally, the leaves of T are the singletons of V_C, that is, the components of the CCG corresponding to the k-PFD. Clearly, by construction each node of T, will have either 0 or 2 children, thus T is full. Moreover, T can be assumed to be ordered, that is, we distinguish the left and the right child of each node. According to Lemma 3.4, every 2-PFD separates a weakly connected component into two child components, connected only in a single direction. In view of this property, we assign to the left child of each node in T, the child component from which the edges originate, and to the right child of the node in T, the component to which the edges terminate.

Our aim is to identify those subsets of edges $\bar{E}_i \subseteq E_i$, that is, sets of edges aligned to a spatial direction $\hat{d}_i$, whose removal gives rise to a k-PFD of the model. The following theorem serves as a characterization of this property.

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

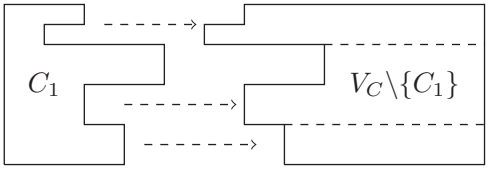

FIGURE 10. 2-PFD of C_1, $V_C\backslash\{C_1\}$.

Theorem 3.7 *Let $\mathscr{M}$ be a toy model and its associated directed graph $G(V,E)$. Let further $G_C(V_C,E_C)$ be the CCG resulting after the removal of a non-empty set of edges $\bar{E}_i \subseteq E_i$, where E_i is the set of all edges of $G(V,E)$ along the direction $\hat{d}_i$. The removal of the edges $\bar{E}_i$ implies a k-PFD, $k \geqslant 2$ of the model $\mathscr{M}$, if and only if G_C is a DAG.*

Proof If $G_C(V_C,E_C)$ is a DAG, then there exists a topological ordering of its vertices $C_1, C_2, \ldots, C_k$, that is, an ordering such that for all $(C_i, C_j) \in E_C$, $i \leqslant j$ holds. In view of this fact, since C_1 is the first in this ordering, there will be only outgoing edges from the vertices of C_1, to those of $V_C\backslash\{C_1\}$. Hence, according to Lemma 3.4 removal of the edges originating from C_1 and terminating to $V_C\backslash\{C_1\}$ is a 2-PFD (see Figure 10). Now, if we denote by G'_C the subgraph of G_C induced by $V_C\backslash\{C_1\}$, we may note that $C_2, \ldots, C_k$, is a topological order of its vertices. Hence, C_2 can be detached from G'_C through a 2-PFD following a similar procedure as above. Thus, after $k-1$ recursive applications of 2-PFD's, utilizing appropriate subsets of $\bar{E}_i$, we obtain a decomposition of the model $\mathscr{M}$ into k weakly connected components $C_1, C_2, \ldots, C_k$, which is a k-PFD.

Conversely, assume that the removal of the set of edges $\bar{E}_i$, implies a k-PFD of the model and let $G_C(V_C,E_C)$ be the corresponding CCG. As explained in Remark 3.6 a k-PFD of a model can be represented by a full, ordered, binary tree T. Moreover, in view of the way that the left and right children are assigned in each node of T, it is easy to verify that if $(C_i, C_j) \in E_C$, then C_i will appear on T to the left of C_j. Hence, if we order the leafs of T starting from the leftmost one moving to the right, we get a total order $C_1, C_2, \ldots, C_k$, which is clearly a topological ordering of $G_C(V_C,E_C)$. Thus, $G_C(V_C,E_C)$ is acyclic. □

4 Maximal PFD along a spatial direction

In the previous section, a characterization of PFD along a particular spatial direction was given in terms of the absence of cycles on the implied CCG. In this section, we propose a method to derive such a maximal acyclic CCG, as the condensation of the graph resulting after making edges not aligned to the chosen direction, bidirectional. In this respect, we introduce the following definitions.

Definition 4.1 (Maximal PFD) Let $\mathscr{M}$ be a toy model and let $G(V,E)$ be the associated directed graph. The removal of a set of edges $\bar{E}_i \subseteq E_i$, along a spatial direction $\hat{d}_i$, implies a *maximal PFD of the model along* $\hat{d}_i$, if the implied CCG is maximal, that is, any set of edges $\bar{E}'_i$, such that $\bar{E}_i \subseteq \bar{E}'_i \subseteq E_i$, implies the same CCG with $\bar{E}_i$.

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

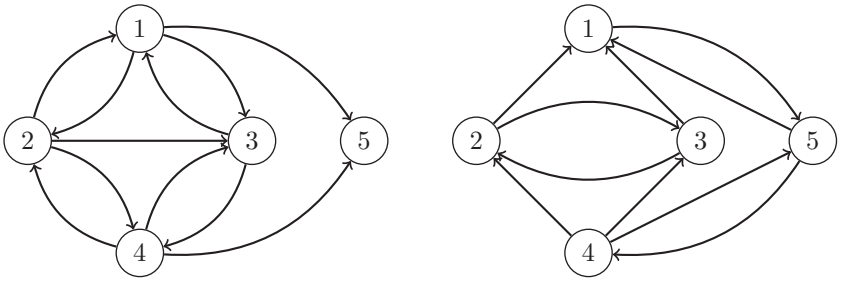

FIGURE 11. The projections of $G(V,E)$ along $\hat{d}_1$ (left), $\hat{d}_2$ (right).

Definition 4.2 (Projection along a direction) Let $\mathscr{M}$ be a toy model, $G(V,E)$ its associated directed graph and let $E_i \subseteq E$ be the set of all edges along the spatial direction $\hat{d}_i$. We define the *projection of* $G(V,E)$ *along the direction* $\hat{d}_i$, to be the graph $G_i(V, E \cup R_i)$, where R_i contains all the edges of G not in E_i, reversed, that is $R_i = \{(u,v) : (v,u) \in E \backslash E_i\}$.

We illustrate the above notion *via* the following example.

Example 4.3 Consider the model presented in Example 3.3 and the corresponding graph $G(V,E)$ shown in Figure 3. According to Definition 4.2, the projection of $G(V,E)$ along $\hat{d}_1$ and $\hat{d}_2$ are shown in Figure 11.

We proceed now to the main result of the present section.

Theorem 4.4 *Let $\mathscr{M}$ be a toy model, let $G(V,E)$ be its associated directed graph* and *let $G^i_{co}(V^i_{co}, E^i_{co})$ be the condensation of the projection $G_i(V, E \cup R_i)$ of $G(V,E)$, along $\hat{d}_i$. Then, $G^i_{co}(V^i_{co}, E^i_{co})$ is a CCG corresponding to a maximal PFD along $\hat{d}_i$.*

Proof Define the set of edges whose endpoints lie on two distinct strongly connected components of $G_i(V, E \cup R_i)$, that is

$$\bar{E}_i = \{(u,v) \in E : u \in C_k, v \in C_l, \text{ where } C_k, C_l \in V^i_{co} \text{ and } k \neq l\}.$$

Note that if either $(u,v) \in E \backslash E_i$ or $(u,v) \in R_i$, then u, v lie on the same strongly connected component of $G_i(V, E \cup R_i)$, because there are edges connecting them in both directions. Thus, $\bar{E}_i \subseteq E_i$.

If any two vertices $u, v \in V$ lie on the same strongly connected component of $G_i(V, E \cup R_i)$, then there exists a directed path from u to v, whose intermediate vertices lie on the same strongly connected component with u, v. Every edge on the path that is in R_i, can be replaced by its 'reverse', which lies in $E \backslash E_i \subseteq E \backslash \bar{E}_i$. The rest of the edges on the path, not in R_i, obviously cannot be in $\bar{E}_i$, since the latter contains edges whose endpoints lie on two distinct strongly connected components of $G_i(V, E \cup R_i)$. Hence, any two vertices $u, v \in V$ lying on the same strongly connected component of $G_i(V, E \cup R_i)$, can be connected *via* an undirected path, which lies entirely on the same weakly connected component as u, v, using

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

only edges from $E \backslash \bar{E}_i$. Thus, all vertices lying on the same strongly connected component of $G_i(V, E \cup R_i)$, belong to the same weakly connected component of $G(V, E \backslash \bar{E}_i)$.

Conversely, if any two vertices $u, v \in V$ lie on the same weakly connected component of $G(V, E \backslash \bar{E}_i)$, then there exists an undirected path from u to v, whose intermediate vertices are on the same weakly connected component with u, v. Our aim is to show that there exists a directed path from u to v in $G_i(V, E \cup R_i)$. In this respect, the edges on the undirected path having the correct orientation, that is from u to v, can be used to form the directed path. If an edge on the undirected path belongs to $E \backslash E_i$ and is oriented from v to u, then it can be replaced in $G_i(V, E \cup R_i)$ by its 'reverse' that belongs to R_i. On the other hand, if an edge on the undirected path belongs to E_i, then both its endpoints must lie in the same strongly connected component of $G_i(V, E \cup R_i)$, otherwise this edge should be in $\bar{E}_i$, whose elements have been removed from $G(V, E \backslash \bar{E}_i)$. In view of this, if such an edge does not have the desired orientation (i.e., from u to v), we can find a directed path in $G_i(V, E \cup R_i)$, with the correct orientation, to replace it. Thus, any two vertices lying on the same weakly connected component of $G(V, E \backslash \bar{E}_i)$, belong to the same strongly connected component of $G_i(V, E \cup R_i)$.

In view of the above discussion, it is clear that the strongly connected components of $G_i(V, E \cup R_i)$ coincide with the weakly connected components of $G(V, E \backslash \bar{E}_i)$. Thus, V_{co}^i is the vertex set of the CCG implied by the removal of the edges of $\bar{E}_i$ from $G(V, E)$. Further, it is straightforward to verify that the set of edges E_{co}^i are exactly the edges of the CCG implied by the removal of the edges of $\bar{E}_i$ from $G(V, E)$. Thus, $G_{co}^i(V_{co}^i, E_{co}^i)$ is a CCG corresponding to the removal of the edges of $\bar{E}_i$. Since the condensation graph of any directed graph is a DAG, the removal of the edges of $\bar{E}_i$, implies a k-PFD of the model, where $k = \left|V_{co}^i\right|$.

To show that the removal of the edges of $\bar{E}_i$, implies a maximal PFD along $\hat{d}_i$, assume there exists a set of edges $\bar{E}_i'$, such that $\bar{E}_i \subseteq \bar{E}_i' \subseteq E_i$, implying a PFD of the model along $\hat{d}_i$. Consider an edge $(u, v) \in \bar{E}_i' \backslash \bar{E}_i$, whose endpoints lie on distinct weakly connected components C_u, C_v, in $G(V, E \backslash \bar{E}_i')$, such that $u \in C_u$ and $v \in C_v$. Clearly, since $(u, v) \notin \bar{E}_i$, it is present in $G(V, E \backslash \bar{E}_i)$ and both u, v lie in the same weakly connected component of the latter. In this case, it is evident from the discussion above that u, v must lie on the same strongly connected component of $G_i(V, E \cup R_i)$. Thus, there exists a directed path from v to u in $G_i(V, E \cup R_i)$. Now, since $u \in C_u$ and $v \in C_v$ in $G(V, E \backslash \bar{E}_i')$, there exists at least one edge $(v', u') \in \bar{E}_i'$, in the directed path from v to u, such that $u' \in C_u$ and $v' \in C_v$, otherwise C_u, C_v would not be distinct. Hence, the weakly connected components C_u, C_v are connected in the CCG implied by the removal of the edges of $\bar{E}_i'$, *via* to opposite edges, which in turn implies that such a removal does not imply a PFD. Having arrived at a contradiction, we conclude that there exists no edge in $\bar{E}_i' \backslash \bar{E}_i$, thus, $\bar{E}_i' = \bar{E}_i$. □

Theorem 4.4 essentially provides a method to obtain a maximal PFD of a given model along a spatial direction $\hat{d}_i$. According to the above result the CCG corresponding to a maximal PFD along $\hat{d}_i$ coincides with the condensation, $G_i(V, E \cup R_i)$, of $G(V, E)$ along this particular direction. Thus, the components into which a maximal PFD decomposes the model, coincide with the strongly connected components of the corresponding projection. The computation of the strongly connected components can be accomplished in linear time, using Kosaraju's algorithm [4,12], Tarjan's strongly connected components algorithm

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

FIGURE 12. The condensations of $G(V, E)$ along $\hat{d}_1$ (left), $\hat{d}_2$ (right).

[14] or Dijkstra's path-based strong component algorithm [5]. Moreover, Kosaraju's and Tarjan's algorithms also compute a reverse topological ordering of the strongly connected components of the graph on which it is applied. The topological ordering computed by these algorithms dictates the order under which the components detected can be detached from the model in the process of a step-by-step decomposition along the chosen spatial direction.

Example 4.5 Applying some strongly connected component computation algorithm on the projections of $G(V, E)$ along $\hat{d}_1$, $\hat{d}_2$, given in Example 4.3, we get respectively the condensations shown in Figure 12. Clearly, the condensed graph corresponding to the projection along $\hat{d}_1$, coincides with the CCG shown in Figure 5, and clearly implies a 2-PFD of the model along this direction. On the other hand, the condensed graph corresponding to the projection along $\hat{d}_2$, consists of only one component, indicating that a k-PFD, for $k \geqslant 2$, along $\hat{d}_2$ is not possible.

5 Hierarchical PFD of toy models and assembly instructions generation

In the present section, an outline of the procedure to obtain a recursive, PFD of a given toy model $\mathscr{M}$ is proposed. The key step of the procedure presented in what follows, is based on both the theoretical analysis presented in Section 3, and the use of well established algorithmic tools for the detection of strongly connected components in directed graphs, as shown in Section 4. While each step of the procedure results in a flat collection of weakly connected components, corresponding to a maximal PFD along some given spatial direction, the outcome of the overall procedure will be a hierarchical model of components, i.e., a rooted tree, having as its top-level component the toy model $\mathscr{M}$ itself, and bottom level elements each of the constituent blocks of the model. Having obtained a hierarchical decomposition of the model, some appropriate tree traversal algorithm may be applied to reverse the decomposition process and produce a step-by-step assembly manual. This procedure is outlined at the end of this section.

Using the setup of the previous sections, assume that $G(V, E)$ is the directed graph associated to the model $\mathscr{M}$. Assume also that each directed edge in E is aligned to one of the p distinct spatial directions $\hat{d}_i$, $i = 1, 2, \ldots, p$. Finally, assume that maximal PFD(C, i) is a readily made function taking as its first argument a weakly connected component of $G(V, E)$ and as its second argument an integer $i = 1, 2, \ldots, p$. The function returns an ordered list of components $C_1, C_2, \ldots, C_k$, $k \geqslant 1$, into which C can be decomposed as the result of a maximal PFD along the direction $\hat{d}_i$. According to the results of Section 4 such a function can be implemented using well-known, linear-time strongly connected components detection algorithms.

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

With this background we define the function HMaxPFD(C), which accepts as argument a weakly connected component of $G(V,E)$, C, and returns a hierarchical decomposition of the model $\mathscr{M}$. The function HMaxPFD is outlined as follows:

HMaxPFD(C)

- Call MaxPFD(C,i) for $i = 1,2,\ldots,p$.
- If for at least one $i = 1,2,\ldots,p$, the number of components $C_1, C_2, \ldots, C_k$, returned by the respective MaxPFD, is greater than 1, then
  - For $j = 1 \ldots k$.
    - — Call AppendChild(C, C_j);
    - — Call HMaxPFD(C_j).

In the above pseudocode, the function AppendChild(C, C_j) is called, which is assumed to append the subcomponent C_j to C, as its child in the hierarchy of the intended decomposition. To implement this in practice, would require each of the discovered components to be able to maintain a list of pointers, pointing from each parent to its children components. The technical details of such an implementation are out of the scope of the present paper. Finally, when the argument of HMaxPFD is a single vertex v (which will necessarily be without edges), we define HMaxPFD(v) = v and the hierarchical operations terminate there, to then pass to the next branch (if any).

To obtain the tree corresponding to the hierarchical PFD of $\mathscr{M}$, with the associated graph $G(V,E)$, one has to invoke the function HMaxPFD, using the entire graph G as its sole argument. To provide a worst case analysis of the complexity of the HMaxPFD algorithm, we first take into account that each run of MaxPFD(C,i) is essentially a call of Tarjan's or a similar algorithm, whose time complexity is $O(|V| + |E|)$, where $|V|, |E|$ are the number of nodes and edges of the graph to which it is applied. Considering the worst case scenario, the MaxPFD will be called at most p times, until an actual decomposition, into two or more subcomponents is obtained. Moreover, at each level of the resulting hierarchical PFD tree, the total number of nodes (blocks) distributed along the components $C_1, C_2, \ldots, C_k$, will be at most n, where n is total number of vertices in G (blocks in $\mathscr{M}$). Thus, if we denote by m the total number of edges in G, then the invocation of MaxPFD(C_j,i), for $i = 1,2,\ldots,p$, $j = 1,2,\ldots,k$ will take at most $O(p(n+m))$ steps. Since the time complexity at each level of the tree is $O(p(n+m))$, the overall worst case complexity will occur on a PFD tree that has the maximum possible height, amongst all the PFD trees with n leaves in total and whose non-leaf nodes have at least two children. This becomes evident if we take into account the fact that the leaves of a PFD tree are exactly the components of G that can be no further decomposed, i.e., its individual blocks. In view of this, the maximum height PFD tree, will be a binary tree where every non-leaf node has exactly two children, out of which at least one is a leaf. The height of such a binary tree with n leaves can be easily seen to be $n-1$. Thus, the worst case time complexity is $O(np(n+m))$.

We illustrate the above procedure in the following example.

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

Example 5.1 Consider the toy model of Example 3.3 and its associated graph shown in Figure 3. Invoking HMaxPFD(G), the procedure will execute as follows:

- Calling MaxPFD$(G, 1)$ returns two components C_1, C_2 where C_1, C_2 consist of the vertices $\{1, 2, 3, 4\}$ and $\{5\}$, respectively.
- Since MaxPFD returned more than one component for $i = 1$,
- For $j = 1$,
- C_1 is appended as a child of G.
- HMaxPFD(C_1) is called.
- MaxPFD$(C_1, 2)$ ($\hat{d}_2$ is the only direction available) returns three components, C_{11}, C_{12} and C_{13}, having as vertex sets $\{1\}$, $\{2, 3\}$ and $\{4\}$, respectively.
- Since MaxPFD returned more than one component for $i = 2$,
- For $j' = 1$,
- C_{11} is appended as a child of C_1.
- HMaxPFD(C_{11}) is called, returning C_{11} since this is a single vertex. Recursion terminates.
- For $j' = 2$,
- C_{12} is appended as a child of C_1.
- HMaxPFD(C_{12}) is called.
- MaxPFD$(C_{12}, 1)$ ($\hat{d}_1$ is the only direction available here) returns two components, C_{121} and C_{122}, having as vertex sets $\{2\}$ and $\{3\}$, respectively.
- Since MaxPFD returned more than one component for $i = 1$,
- For $j'' = 1$,
- C_{121} is appended as a child of C_{12}.
- HMaxPFD(C_{121}) is called, returning C_{121} since this is a single vertex. Recursion terminates.
- For $j'' = 2$,
- C_{122} is appended as a child of C_{12}.
- HMaxPFD(C_{122}) is called, returning C_{122} since this is a single vertex. Recursion terminates.
- For $j' = 3$,
- C_{13} is appended as a child of C_1.
- HMaxPFD(C_{13}) is called, returning C_{13} since this is a single vertex. Recursion terminates.
- For $j = 2$,
- C_2 is appended as a child of G.
- HMaxPFD(C_2) is called, returning C_2 since this is a single vertex. Recursion terminates.

The resulting hierarchical PFD of the model is depicted in Figure 13. The assembly instructions for the model can be recovered by applying a depth-first traversal, starting from the root node of the tree.

Having obtained a hierarchical decomposition of a toy model $\mathscr{M}$, which is essentially a tree structure like the one shown in Figure 13, we can proceed to the composition of

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

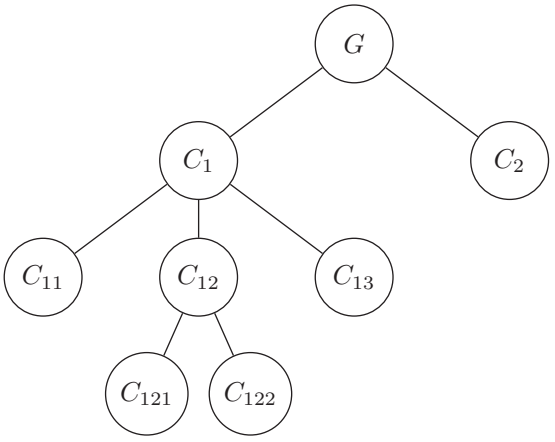

FIGURE 13. The hierarchical PFD of the toy model in Example 3.3.

its nodes to reverse the PFD and produce the assembly instructions. This goal can be accomplished by employing a tree traversal algorithm, which respects the parent–child hierarchy, in the sense that each node is visited after its children. The necessity of the requirement regarding the priority of visits between parents and their children, emerges from the fact in order to assemble a component, from its constituent subcomponents, i.e., the children of the node in the tree, one has to assemble each child component first.

An algorithm appropriate for this task could be a postorder traversal applied on the hierarchical PFD tree of $\mathscr{M}$. Preorder, inorder and postorder are well-known traversal procedures that can be applied on ordered binary trees, i.e., rooted trees whose nodes have at most two children labelled as 'left' and 'right'. A preorder traversal visits first the parent node, then traverses the left subtree and finally the right subtree. Respectively, the inorder traversal first traverses the left subtree of a node, then visits the node itself and finally traverses the right subtree. Postorder traversal, traverses first the left subtree of a node, then its right subtree and finally it visits the node itself.

While inorder traversal may be ambiguous when applied to a general (non-binary) ordered tree, preorder and postorder traversals are well defined. Here, we focus on the generalized version of the postorder traversal algorithm, which is applicable to non-binary trees. Given a node p in such a tree, the postorder traversal procedure can be defined recursively as follows:

- Traverse the leftmost child of p.
- Visit the node p.
- Traverse the right sibling of p.

The output of such an algorithm is a series of nodes ordered in such a way that parent nodes appear in the sequence after all their children. In view of this fact, the sequence generated by the postorder traversal can be used to generate the assembly instructions of the toy model $\mathscr{M}$. It should be noted that the leaf nodes of the hierarchical PFD tree represent individual toy blocks that require no assembly, thus they can be safely neglected in the instructions generation procedure. On the other hand, the internal nodes of the tree

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

represent components of the toy model consisting of an assembly of individual blocks or other subcomponents, and thus they are the ones for which assembly instructions are needed. The ingredients required for the assembly of those components are no other than their child nodes in the hierarchy. If a subcomponent is used as a building block for a higher level component in the PFD hierarchy, then the former will precede the latter in the ordered sequence produced by the postorder traversal. Moreover, as noted in Section 1, the exact geometry of the interconnections between any two blocks has been recorded beforehand. Thus, identifying the blocks from which a component is comprised, provides enough information to recover the exact geometric structure of each component. As a result, following the order dictated by the traversal, the assembly instructions of every subcomponent comprising a higher level component will appear earlier in the assembly procedure manual, leaving no room for inconsistencies in the flow of instructions.

The procedure is illustrated in the following example.

Example 5.2 Given the hierarchical PFD of Example 5.1, we may apply the postorder traversal procedure on the tree shown in Figure 13. The outcome of the algorithm is the following sequence of nodes:

$$C_{11}, C_{121}, C_{122}, C_{12}, C_{13}, C_1, C_2, G.$$

As explained above the leaf nodes C_{11}, C_{121}, C_{122}, C_{13} and C_2 can be safely omitted, since they require no assembly. Doing so, the sequence of the remaining nodes consists only of C_{12}, C_1 and G, in that particular order.

Thus, the assembly instructions manual in this case should consist of the following three steps:

(1) Show how C_{12} is assembled from its children nodes C_{121}, C_{122}.

(2) Respectively, show how C_1 can be assembled from C_{11}, C_{12}, C_{13}.

(3) Finally, show how G is assembled from using C_1, C_2.

6 Conclusions

In this note, we study the problem of automatically producing step-by-step assembly instructions for Engino® toy models. The assembly manual of a toy model can be generated by reversing the decomposition process of the model to its constituent blocks. As explained in Section 2, the disassembly process may under certain circumstances be blocked due to the presence of particular geometric structures in the interconnections between blocks. To avoid such situations we propose a graph-theoretic framework for the analysis of the problem and provide a characterization of the decompositions that are physically feasible. Moreover, a procedure to obtain maximal PFDs along a given geometric direction is presented, which can be implemented using well known, linear-time algorithms for the detection of strongly connected components in directed graphs. Based on these results, an algorithmic procedure for the hierarchical decomposition of a given toy model, which takes into account the physical feasibility of the intermediate steps, is proposed. The final goal of generating a sequence of assembly instructions for the model

https://doi.org/10.1017/S0956792518000086 Published online by Cambridge University Press

is accomplished, by applying a postorder traversal of the hierarchical decomposition tree, from which a step-by-step series of instructions can be easily recovered.

As for future extensions that could stem from our presented approach, and future challenges to be tackled, we remark the following:

- First of all, notice that the connection principle for ENGINO blocks is mainly of binary type (just like those of LEGO and other toy systems), in the sense that even though some types of blocks have several male and/or female connectors, thus allowing for several ways of connecting two given blocks, any connection between two blocks is achieved by matching at least one pair of male-female connections, resulting in a finite set of possible relative spatial configurations between the blocks. An important exception to this is the freely-rotating pivoting connection, which allows for a continuous choice of the pivot angle, so the set of relative spatial configurations becomes infinite. In this paper, we have focused on the binary type of connections because of the resulting finiteness of the set of possible spatial configurations, which allows us to tackle the problem by defining the connection directions $\hat{d}_i, \quad i = 1, \ldots, p$. In future work, the feature of pivoting connections will be added to our PFD, based on the fact that pivoting connections during the assembly process must be geometrically feasible, in the sense that small displacements associated with the rotation degree of freedom about the pivot point must be allowed to happen. The main difficulty lies in extending the current definition of the 'fixed' connection directions $\hat{d}_i, \quad i = 1, \ldots, p$, which will have to depend on the continuous 'pivoting' degrees of freedom.
- From the previous point it follows that our method can be applied to several toy systems, and even beyond that to industrial assembly processes [10,15] with binary-type connections as defined above. The main advantage of our method is that it requires very little geometrical and physical information about the connecting pieces. This is, at the same time, the main limitation of the method. For example, it does not apply to assemblies that require three or more hands [13], and more generally it does not deal with cases when force and torque balances are relevant, as in the problems of grasping parts (form closure, force closure, etc.) [16]. However, this does not mean our method cannot be used in combination with these and other advanced assembly features. In fact, our method could be included as a complementary module in dis-assembly process planning for existing products in industry. For example, the feature of linearizability [16], of practical importance in assembly lines, could be incorporated into our method because it is related to the distribution of internal nodes and leaves in our hierarchical PFD graphs. And, with a little bit of imagination, our method could potentially find its utility as a module in the recently discovered molecular assembly processes [7], because these processes are characterized by constrained geometric arrangements, local interactions and reduced reactivity.

Acknowledgements

This work originated from our participation in the 125th European Study Group with Industry (1st Study Group with Industry in Cyprus). We thank the Mathematics for

Industry Network (MI-NET, www.mi-network.org), COST Action TD1409 for generous funding and support with the logistics of this first Study Group with Industry in Cyprus.

We would also like to thank Costas Sisamos, founder and CEO of Engino Ltd, for the detailed exposition of the problem, his valuable insight on it and his comments on the present paper. Finally, we would like to thank the editor and the anonymous referees for taking time in reading and suggesting modifications to the paper. We highly appreciate it, as the comments have been very useful in improving the paper.

References

[1] Agrawala, M. *et al.* (2003) Designing effective step-by-step assembly instructions. In: *Proceedings of ACM SIGGRAPH 2003 ACM Transactions on Graphics (TOG)*, Vol. 22, pp. 828–837.

[2] Bang-Jensen, J. & Gutin, G. Z. (2009) *Digraphs: Theory, Algorithms and Applications*, 2nd ed., Springer-Verlag London.

[3] Bondy, J. A. & Murty, U. S. R. (1976) *Graph Theory with Applications*. Vol. 290. London: Macmillan.

[4] Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2001) *Introduction to Algorithms*, 2nd ed., The MIT Press, Cambridge, Massachusetts, and McGraw-Hill Book Company, Boston.

[5] Dijkstra, E. (1976) *A Discipline of Programming*, Prentice Hall, NJ.

[6] Hsu, Y.-Y., Tai, P.-H., Wang, M.-W. & Chen, W.-C. (2011) A knowledge-based engineering system for assembly sequence planning. *Int. J. Adv. Manuf. Technol.* **55**, 763–782.

[7] Kassem, S., Lee, A. T., Leigh, D. A., Marcos, V., Palmer, L. I. & Pisano, S. (2017) Stereodivergent synthesis with a programmable molecular machine. *Nature* **549**, 374–378.

[8] Lambert, A. J. D. (2003) Disassembly sequencing: A survey. *Int. J. Prod. Res.* **41**, 3721–3759.

[9] Li, W., Agrawala, M., Curless, B. & Salesin, D. (2008) Automated generation of interactive 3D exploded view diagrams. In: *Proceedings of ACM SIGGRAPH 2008 ACM Transactions on Graphics (TOG)*, Vol. 27, p. 101.

[10] Natarajan, B. K. (1988) On planning assemblies. In: *Proceedings of the 4th Annual Symposium on Computational Geometry*, ACM, pp. 299–308.

[11] Peysakhov, M., Galinskaya, V. & Regli, W. C. (2000) Representation and evolution of lego-based assemblies. In: *Proceedings of the AAAI/IAAI*, p. 1089.

[12] Sharir, M. (1981) A strong connectivity algorithm and its applications to data flow analysis. *Comput. Math. Appl.* **7**, 67–72.

[13] Snoeyink, J. & Stolfi, J. (1994) Objects that cannot be taken apart with two hands. *Discret. Comput. Geom.* **12**, 367–384.

[14] Tarjan, R. E. (1972) Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1**, 146–160.

[15] Wang, L., Keshavarzmanesh, S., Feng, H. Y. & Buchal, R. O. (2009) Assembly process planning and its future in collaborative manufacturing: A review. *Int. J. Adv. Manuf. Technol.* **41**, 132–144.

[16] Wilson, R. H. & Latombe, J. C. (1994) Geometric reasoning about mechanical assembly. *Artif. Intell.* **71**, 371–396.