Increasing the Creativity of ENGINO Toy Sets and Generating Automatic Building Instructions

A. Araújo, CMUC, Department of Mathematics, University of Coimbra, Portugal

A. Gibali, Department of Mathematics, ORT Braude College, Israel

A. Kyprianou, Department of Mechanical and Manufacturing Engineering, University of Cyprus

E. Antoniou,[*] Department of Information Technology, Alexander Technological Educational Institute of Thessaloniki, Greece

M. D. Bustamante, Institute for Discovery, School of Mathematics and Statistics, University College Dublin, Ireland

Y. Kaminski, Applied Mathematics Department, Holon Institute of Technology, Israel

W. Okrasinski, University of Technology, Institute of Mathematics and Computer Science, Wroclaw, Poland

Team participants
A. Araújo, A. Gibali, A. Kyprianou, E. Antoniou,
M. Bustamante, Y. Kaminski, W. Okrasinski, G. Benhame,
A. Riseth, C. Morosanu, I. Porumbel, C. Deliyiannis,
A. Micheletti, P. Hjorth, H. Ockendon

The problem was presented by
Costas Sisamos, Founder and General Director, ENGINO Ltd

at the **125th European Study Group with Industry (ESGI125)**
(1st Study Group with Industry in Cyprus, www.esgi-cy.org)

Abstract

During the First Study Group with Industry which was held in Limassol, Cyprus, the ENGINO® TOY SYSTEM introduced two challenging problems. The first is to get bounds on the number of possible models/toys which can be constructed using a given package of building blocks. And the second is to generate automatically the assembly instructions for a given toy. In this report we summarize our insights and provide preliminary results for the two challenges.

KEYWORDS: 05A05, 05C20, 05C40

[*]Corresponding author.

1 Introduction

ENGINO$^{®}$ TOY SYSTEM was founded by Costas Sisamos in 2004 in order to commercialize his invention of a new system of multi-functional plastic connectors. After successfully receiving research funding, Costas left his full-time job as an educator to fully engage in R&D, and after 3 years of designing, prototyping and testing, in 2007 he launched the first sets of ENGINO construction toys. The ENGINO toys are created by assembling small pieces 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. The products grasped the attention of global buyers and toy experts and by now more than 60 different toy sets are manufactured in his production facility in Limassol, Cyprus, covering various age levels and price ranges, from simple sets to solar energy and robotics. The company has experienced steady growth, reaching presence in more than 40 countries by now.

Each of the 60 toy sets produced by ENGINO has a specific number of blocks that can be assembled into many different models. Based on experience, it has been observed that the creative potential of the system increases geometrically as the number of blocks in the set increase. This is due to the patented design of the ENGINO blocks that allow connectivity from many directions simultaneously.

2 Description of the challenges

For this 125th European Study Group with Industry, that took place in Limassol, Cyprus, from 5-9 December 2016, the company proposed two challenges.

1. Given a package of blocks, evaluate the number of possible models/toys using the package's blocks. With this “creativity measure" the company is able to know if the sets have indeed been optimized or if more models are also possible. Also, by substituting some blocks with others maybe the level of creativity can be increased. Furthermore it will provide a marketing tool to explain the creativity of the system.

2. Another big challenge for the company is to be able to generate the assembly instructions for each toy automatically. For most of the toy sets the instructions are currently being created manually. The developers attempted to create an automatic disassembly module in a proprietary 3D builder software which is based on the UNITY 3D-Game Engine. However, the system cannot predict which block needs to be connected first during the assembly instructions. The priority of parts is random and incorrect, making the generated instructions not usable.

   It is important to find a solution to this problem and have an algorithm that will be able to prioritize the different parts of the structure or substructure, which we can feed in the software so that we shall optimize the de-structuring of a model with physically correct priority sequence. There may be endless possibilities that can work so a solution may seem impossible, however there some solutions that definitely will not work and those are the ones we need to be able to identify and remove from the possible assembly sequence.

The main goal of this report is to give an answer to second challenge. Nevertheless, we will give a partial answer for the first one, considering the particular case where we just have constructions obtained by linearly assembling the blocks. We may define a *linear assembly* construction as a model obtained by an ordered sequence of blocks such that each block is connected only to the previous (if it exists) and to the following (if it exists) ones.

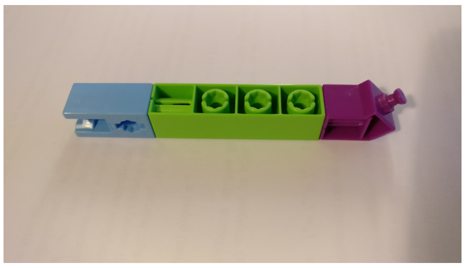


Figure 1: Linear assembly

Theorem 1. *Given a set of n blocks $(B_{i,1},\dots,B_{i,n})$, the number of possible linear assemblies with this set is given by*

$$N = \frac{1}{t_1!\dots t_k!} \sum_{1\le i_1<\dots<i_n\le n} v(B_{i_1})\, v(B_{i_n}) \prod_{j=2}^{n-1} \big(v(B_{i_j})-1\big)\,,$$

where t_j is the number of blocks of type j and $v(B_{i_j})$ is the valency of a block B_{i_j}, defined as the maximal number of possible connections of B_{i_j}.

Proof. The first piece offers a number of connections equal to its valency. Other pieces can be connected to the previous one in a number of ways equal to their valencies. Except the last one, they also offer a number of possible connections to the following piece equal to their valencies minus one.

□

During the meeting P. Hjorth and H. Ockendon proposed a formulation of optimization problem whose solution could help the company build packages able to produce a maximal number of models under given constraints. The optimization model was developed by A. Riseth and its results can be found in Appendix A of the present report.

3 Mathematical model and analysis

Let us now focus on the second challenge. As we mentioned before, currently, for most toys, the instructions are being created manually. Our goal is to develop an automatic assembly instruction manual for each toy. To do this we actually follow the reverse process, that is, given a description of a toy model, which is available in the company's database, we develop a method to check whether a disconnection between particular blocks of the model is physically possible. In what follows we call this procedure a *Physically Feasible Decomposition* (PFD) of the model. The result of such a decomposition would be a collection of sub-models, on which the method can be recursively applied until no further decompositions are possible. Once the model has been decomposed to its constituent blocks, the steps of the decomposition can be reversed to produce its assembly instructions manual.

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 M, we associate to it a directed graph $G(V,E)$, where

- $V=\{v_1,v_2,\dots,v_n\}$ is the vertex set of G with each vertex v_i corresponding to a block of M,
- $E=\{(u,v) : u\in\mathrm{V}, v\in\mathrm{V}\}$ 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 $\hat{\imath}, \hat{\jmath}, \hat{k}$ 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 with $\hat{d}$.

Assuming that all the connections of the model M correspond to p distinct spatial directions, we can partition the edge set E into a family of p mutually disjoint sets E_i, $i = 1, 2, \dots, p$, each of which contains the edges associated to connections sharing the same direction in space.

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

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

Example 1. Consider the two blocks shown in the following figure (Fig. 2)

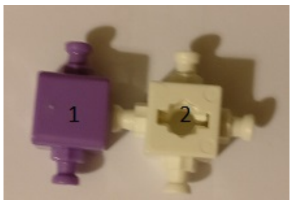


Figure 2: Two blocks that can be disconnected

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 the following picture (Fig. 3)

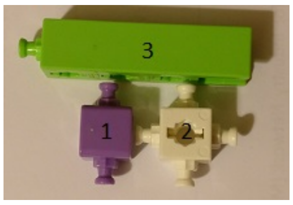


Figure 3: Blocks $1, 2$ cannot be disconnected

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.

Our aim is to identify a set of connections between the blocks of the model that can be disconnected simultaneously without violating the Principle of Physically Feasible Disconnection stated above. We call this a *Physically Feasible Decomposition* (PFD) of the model. The problem of finding a PFD of a toy model can be stated as follows:

Problem 1. Find a subset of edges $\bar{E}_i \subseteq E_i$, for some $i = 1, 2, \dots, p$, whose removal imply a PFD of the model into two or more submodels.

To obtain a PFD we propose the following procedure:

1. Choose a subset of edges $\bar{E}_i \subseteq E_i$, whose connections are aligned in the same spatial direction.
2. Remove all edges of $\bar{E}_i$ from G and all directions from edges to obtain the undirected graph $G'(V, E \setminus \bar{E}_i)$.
3. Apply some connectivity search algorithm (see Algorithm 1) on G' to identify its connected components C_j, $j = 1, 2, \dots, k$.
4. For each $(s, t) \in \bar{E}_i$, identify the pair of components (C_s, C_t) such that $s \in C_s$ and $t \in C_t$.
5. Considering the connected components C_j, $j = 1, 2, \dots, k$, as vertices and the (distinct) pairs of components resulting from step 4 as directed edges, create the directed *Components Connectivity Graph* (CCG). We denote the directed CCG by $G_C(V_C, E_C)$, where $V_C = \{C_1, C_2, \dots, C_k\}$ and E_C is the set of (distinct) pairs of components resulting from step 4 as directed edges.

```
Input: Undirected version of the graph $G(V,E)$ and a subset of edges $\bar{E}_i \subseteq E$ on a given
       direction to be removed.
Output: Connected Components $C_1, C_2, \dots, C_k$.
$E' \leftarrow E \setminus \bar{E}_i$
$V' \leftarrow V$
$k \leftarrow 0$
while $V' \neq \emptyset$ do
    Pick an $s \in V'$
    $C_k \leftarrow DFS(G(V,E'), s)$          // Run a Depth First Search starting
                                              // from vertex s.  The result
                                              // is a connected component
    $V' \leftarrow V' \setminus C_k$
    $k \leftarrow k + 1$
end
```

Algorithm 1: Detection of connected components

The following theorem provides a criterion to decide whether the removal of a set of edges along a given direction gives rise to PFD of the model.

Theorem 2. *Let 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 subset of edges $\bar{E}_i \subseteq E_i$, where E_i is the set of all edges of $G(V, E)$ along the direction $\hat{d}_i$. If G_C contains no directed cycles, then the removal of the edges $\bar{E}_i$ implies a PFD of the model M.*

Proof. We shall use induction on the number of components, in $V_C = \{C_1, C_2, \dots, C_k\}$.

- If $k = 1$ and there are no directed cycles, the CCG can have no edges. Thus, no disconnections between blocks take place and the component C_1 can be trivially considered as a PFD of the model, since the PFD principle is not violated.
- Assuming that the theorem holds for any CCG with k components, where $k \geq 1$, we shall prove it for any CCG with $k + 1$ components.

Let $\bar{G}_C(\bar{V}_C, \bar{E}_C)$ be a CCG, with $k+1$ components which contains no directed cycles. We first show that if $\bar{G}_C$ contains no directed cycles, then there exists a vertex (component) $\bar{C}_1 \in \bar{V}_C$, with no incoming edges. Since $\bar{G}_C$ has no directed cycles, all its paths will be of finite length. Thus, let the sequence $P = (\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_m)$, $m \leq k+1$, be a path of maximal length in $\bar{G}_C$. Clearly, if there was an incoming edge on $\bar{C}_1$, there should be a vertex $\bar{C}_0 \in \bar{V}_C$ connected to $\bar{C}_1$ through the edge $(\bar{C}_0, \bar{C}_1)$. In such a case the path $P' = (\bar{C}_0, \bar{C}_1, \bar{C}_2, \ldots, \bar{C}_m)$, would be longer than P, which has been assumed to be maximal. Thus, $\bar{C}_1$ has no incoming edges.

Using this fact, since $\bar{C}_1$ has only outgoing edges, the underlying physical connections between $\bar{C}_1$ and the rest of the components of $\bar{V}_C$, will point towards the direction $\hat{d}_i$. In other words, the component $\bar{C}_1$ is connected to the rest of the model only on the one side, leaving its other side free. Thus, applying opposite forces on $\bar{C}_1$ and the rest of the model, will result in a PFD of the model, since $\bar{C}_1$ is free to move towards the direction $-\hat{d}_i$ (see Fig. 4).

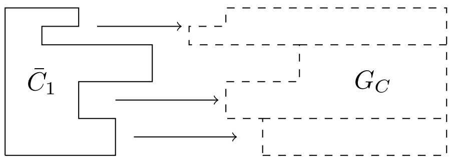


Figure 4: Physical disconnection of $\bar{C}_1$ from G_C

By detaching the component $\bar{C}_1$ from the model, the remaining components of $\bar{G}_C(\bar{V}_C, \bar{E}_C)$ form a CCG $G_C(V_C, E_C)$, consisting of k components with no directed cycles. Hence, the induction hypothesis applies to $G_C(V_C, E_C)$ and implies that a PFD of the submodel corresponding to $G_C(V_C, E_C)$ is possible. Combining the physically feasible disconnection of component $\bar{C}_1$, with the PFD implied by the induction hypothesis for the submodel corresponding to $G_C(V_C, E_C)$, we obtain a PFD of the entire model M.

□

The key idea behind the proof of the above theorem is the well known fact (see for instance [1, 2]) that every directed acyclic graph has a topological ordering.

According to the above theorem if the CCG of a model contains no directed cycles then a PFD is implied. In the case where directed cycles are present in a CCG, we may easily eliminate them by removing all the edges of $\bar{E}_i$ that give rise to edges of E_C lying on the directed cycles of the CCG. The resulting CCG will no longer have directed cycles, since the constituent components of each cycle will collapse to a single component. Thus, according to the theorem this new CCG will imply of PFD of the model. Moreover, if the new CCG contains two or more components the implied PFD will be a non trivial one.

4 Examples

Let us now consider three illustrative examples.

Example 2. Consider the following model (Fig. 5)

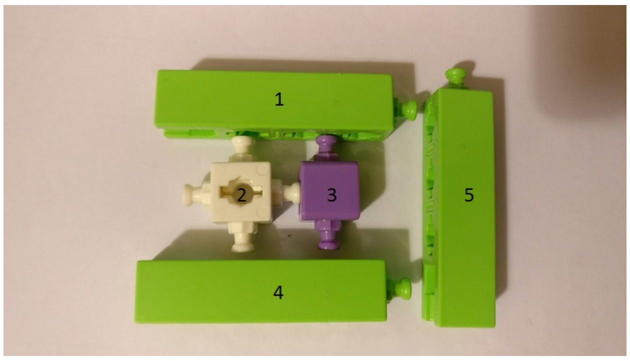


Figure 5: Model 1

and the corresponding graph

$$\begin{tikzcd}
& 1 \arrow[rrd, "\hat{\imath}"] & & \\
2 \arrow[ru, "\hat{\jmath}"] \arrow[rr, "\hat{\imath}"] & & 3 \arrow[lu, "\hat{\jmath}"'] & 5 \\
& 4 \arrow[lu, "\hat{\jmath}"] \arrow[ru, "\hat{\jmath}"'] \arrow[rru, "\hat{\imath}"'] & &
\end{tikzcd}$$

Case 1 Removal of edges in the direction $\hat{\imath}$:

The edges to be removed are $(2, 3)$, $(1, 5)$, $(4, 5)$. After the removal, the graph becomes

$$\begin{tikzcd}
& 1 & & \\
2 \arrow[ru, "\hat{\jmath}"] & & 3 \arrow[lu, "\hat{\jmath}"'] & 5 \\
& 4 \arrow[lu, "\hat{\jmath}"] \arrow[ru, "\hat{\jmath}"'] & &
\end{tikzcd}$$

in which the following two components can be identified

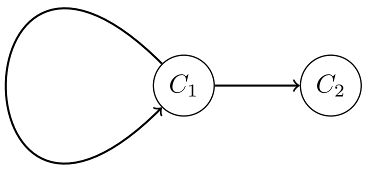
$$C_1 = \{1, 2, 3, 4\}$$

$$C_2 = \{5\}$$

Next, the CCG is formed by associating to each of the removed edges an edge between the components which it was connecting, i.e.

$$\begin{aligned}(2,3) &\longrightarrow (C_1, C_1)\\(1,5) &\longrightarrow (C_1, C_2)\\(4,5) &\longrightarrow (C_1, C_2)\end{aligned}$$

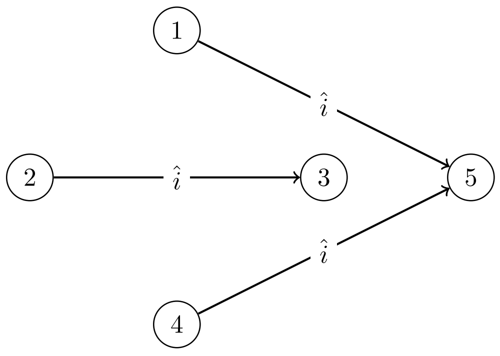
Thus, the CCG $G_C(V_C, E_C)$ is



with $V_C = \{C_1, C_2\}$ and $E_C = \{(C_1, C_1), (C_1, C_2)\}$. According to Theorem 2, the edges participating to the loop on C_1 are not physically removable, hence the only PFD can be obtained from the edges (1,5), (4,5).

Case 2 Removal of edges in the direction $\hat{j}$:

The edges to be removed are $(2, 1)$, $(4, 2)$, $(3, 1)$, $(4, 3)$. After the removal, the graph becomes



with the following components

$$C_1 = \{2, 3\}$$
$$C_2 = \{1, 4, 5\}$$

The CCG is formed by associating to each of the removed edges an edge between the components which it was connecting, i.e.

$$\begin{aligned}(2,1) &\longrightarrow (C_1, C_2)\\(4,2) &\longrightarrow (C_2, C_1)\\(3,1) &\longrightarrow (C_1, C_2)\\(4,3) &\longrightarrow (C_2, C_1)\end{aligned}$$

Thus, the CCG is

$$\begin{tikzcd}C_1 \arrow[r, bend left] & C_2 \arrow[l, bend left]\end{tikzcd}$$

All removed edges are participating the directed cycle between C_1 and C_2, no disconnection is physically feasible in this direction.

Example 3. Consider the following model

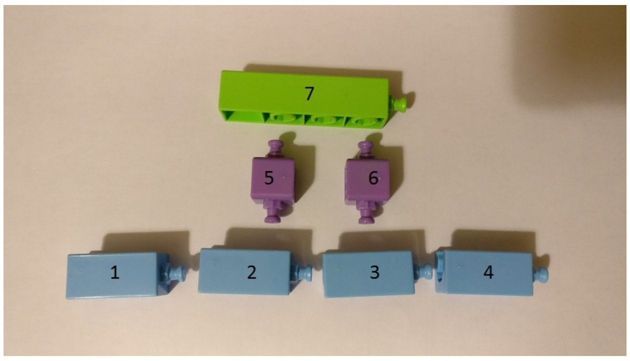


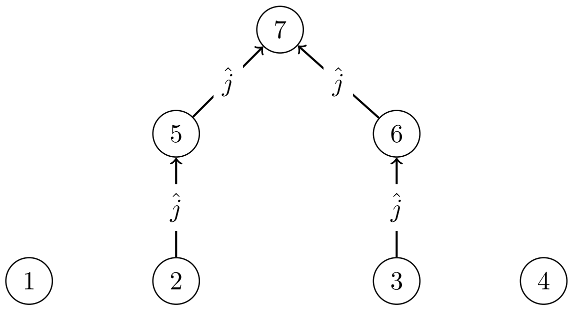
Figure 6: Model 2

and the corresponding graph

$$\begin{tikzcd}
& & 7 & & \\
& 5 \arrow[ur, "\hat{j}"] & & 6 \arrow[ul, "\hat{j}"'] & \\
1 \arrow[r, "\hat{i}"] & 2 \arrow[u, "\hat{j}"] \arrow[rr, "\hat{i}"] & & 3 \arrow[u, "\hat{j}"] \arrow[r, "\hat{i}"] & 4
\end{tikzcd}$$

Case 1 Removal of edges in the direction $\hat{i}$:

The edges to be removed are $(1,2)$, $(2,3)$, $(3,4)$. After the removal, the graph becomes



in which the following components can be identified

$$C_1 = \{1\}$$

$$C_2 = \{2, 3\}$$
$$C_3 = \{4\}$$

The CCG is formed by associating to each of the removed edges an edge between the components which it was connecting, i.e.

$$\begin{aligned}(1,2) &\longrightarrow (C_1, C_2)\\ (2,3) &\longrightarrow (C_2, C_2)\\ (3,4) &\longrightarrow (C_2, C_3)\end{aligned}$$

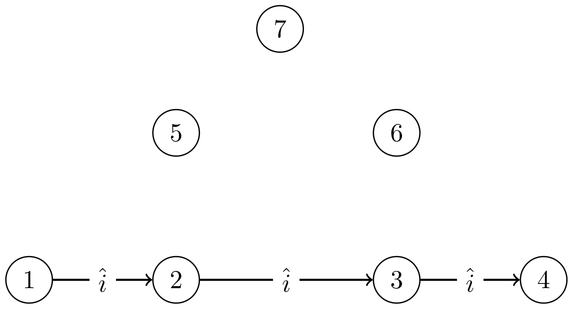
Thus, the CCG is

$$\begin{tikzcd} C_1 \arrow[r] & C_2 \arrow[r] \arrow[loop above] & C_3 \end{tikzcd}$$

The edges participating to the loop on C_2 are not physically removable, hence the only physically feasible disconnection can be obtained from the edges (1,2), (3,4).

Case 2 Removal of edges in the direction $\hat{j}$:

The edges to be removed are $(2,5)$, $(5,7)$, $(3,6)$, $(6,7)$. After the removal, the graph becomes



in which the following components can be identified

$$C_1 = \{1, 2, 3, 4\}$$

$$C_2 = \{5\}$$
$$C_3 = \{6\}$$
$$C_4 = \{7\}$$

The CCG is formed by associating to each of the removed edges an edge between the components which it was connecting, i.e.

$$\begin{aligned}(2,5) &\longrightarrow (C_1, C_2)\\ (5,7) &\longrightarrow (C_2, C_4)\\ (3,6) &\longrightarrow (C_1, C_3)\\ (6,7) &\longrightarrow (C_3, C_4)\end{aligned}$$

Thus, the CCG is

$$\begin{tikzcd}
& C_4 & \\
C_3 \arrow[ur] & & C_2 \arrow[ul] \\
& C_1 \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Since there is not a directed cycle in the connectivity graph all edges are physically removable.

Example 4. Consider the following model

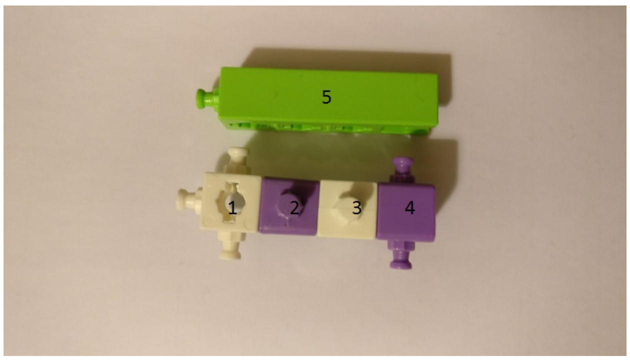


Figure 7: Model 3

the corresponding graph

$$\begin{tikzcd}
 & & & 5 & & & \\
1 \arrow[urrr, "\hat{\jmath}"] \arrow[rr, "\hat{\imath}"] & & 2 \arrow[rr, "\hat{\imath}"] & & 3 \arrow[rr, "\hat{\imath}"] & & 4 \arrow[ulll, "\hat{\jmath}"']
\end{tikzcd}$$

Case 1 Removal of edges in the direction $\hat{\imath}$:

The edges to be removed are $(1,2)$, $(2,3)$, $(3,4)$. After the removal, the graph becomes

$$\begin{tikzcd}
 & & & 5 & & & \\
1 \arrow[urrr, "\hat{\jmath}"] & & 2 & & 3 & & 4 \arrow[ulll, "\hat{\jmath}"']
\end{tikzcd}$$

in which the following components can be identified

$$C_1 = \{1, 4, 5\}$$

$$C_2 = \{2\}$$
$$C_3 = \{3\}$$

The CCG is formed by associating to each of the removed edges an edge between the components which it was connecting, i.e.

$$\begin{aligned}
(1,2) &\longrightarrow (C_1, C_2) \\
(2,3) &\longrightarrow (C_2, C_3) \\
(3,4) &\longrightarrow (C_3, C_1)
\end{aligned}$$

Thus, the CCG is

$$\begin{tikzcd}
 & C_1 \arrow[dl] & \\
C_2 \arrow[rr] & & C_3 \arrow[ul]
\end{tikzcd}$$

All edges participate the directed cycle between C_1, C_2, C_3, hence none of them is physically removable.

Case 2 Removal of edges in the direction $\hat{\jmath}$:

The edges to be removed are $(1,5)$, $(4,5)$. After the removal, the graph becomes

$$\begin{tikzcd}
 & & & 5 & & & \\
1 \arrow[rr, "\hat{\imath}"] & & 2 \arrow[rr, "\hat{\imath}"] & & 3 \arrow[rr, "\hat{\imath}"] & & 4
\end{tikzcd}$$

in which the following components can be identified

$$C_1 = \{1, 2, 3, 4\}$$

$$C_2 = \{5\}$$

The CCG is formed by associating to each of the removed edges an edge between the components which it was connecting, i.e.

$$\begin{aligned}(1, 5) &\longrightarrow (C_1, C_2)\\ (4, 5) &\longrightarrow (C_1, C_2)\end{aligned}$$

Thus, the CCG is

$$\begin{tikzcd} C_1 \arrow[r] & C_2 \end{tikzcd}$$

Since there are no directed cycles, all edges are physically removable.

5 Conclusions and recommendations to the company

A systematic procedure for the Physically Feasible Decomposition (PFD) of an ENGINO® TOY SYSTEM model has been proposed. A directed graph using as vertices the blocks of the model, and as edges, connections between them, is used to capture the structure of the model. The edges of the graph are labeled with a direction vector used to identify geometric direction of the underlying connection between blocks on the original model. Removing groups of edges sharing the same direction vector, we identify the resulting connected components of the graph, which in turn are used to construct a higher - level, structure - graph called the Components Connectivity Graph (CCG). The absence of directed cycles in the CCG is in turn shown to play a key role in the determination of the subset of edges whose removal implies a PFD of the model.

We strongly advise ENGINO to do a pilot-test before implementing this model. In order to have an effective implementation, the model will need further developments, tests and implementation issues with respect to its adherence to the company particular needs and related problems. Even though the model yield good preliminary results, it must be validated using more complicated toys. This is a crucial step that can be carried by a short-term project (MSc) or an internship in a close collaboration with the industrial partner. The company should also consider offering a PhD Scholarship and/or a Postdoc Fellowship. We believe that the model here described can provide some useful hints with interest for the optimization of production process.

References

[1] J. Bondy, U. Murty, Graph Theory with Applications, Elsevier Science Publishing Co. Incorporated, 1976.

[2] J. Bang-Jensen and G. Z. Gutin, Digraphs: Theory, Algorithms and Applications, 2nd ed. Springer Publishing Company, Incorporated, 2008.

A Appendix - Optimise Engino box sets

Problem formulation

Given a list of models, we want to create a box set that can create as many of these models within a requirements on cost and complexity. The number of different models will be denoted by M, and the number of different Engino parts by N. The information on how many pieces of each part the models require is stored in a matrix $P \in \mathbb{N}^{M\times N}$. Let the element $P_{i,j}$ denote the number of pieces of part j, required to build model i.

An Engino box set consists of $x_j \in \mathbb{N}$ pieces for each of the $j = 1, \dots, N$ parts. A model i can be created from the box set if $x_j \geq P_{i,j}$ for each part j. If this holds true, we say $y_i = 1$, if not $y_i = 0$. The total number of models we can build with a given box set is then $\sum_{i=1}^{M} y_i$. Each part j has a cost c_j per piece, so the total cost of a box set becomes $\sum_{j=1}^{N} c_j x_j$.

We can thus set up an optimisation problem to create a box set that can maximise the number of models, with a cost less than $C > 0$:

$$\begin{aligned} \max_{\substack{y\in\{0,1\}^M \\ x\in\mathbb{N}^N}} \sum_{i=1}^{M} y_i \quad \text{s.t.} & \\ x_j \geq p_{i,j} y_i \qquad & \forall i,j \\ \sum_{j=1}^{N} c_j x_j \leq C. & \end{aligned} \tag{1}$$

Note that the size of the optimisation problem may be reduced if there are models i such that $\sum_{j=1}^{N} c_j p_{i,j} > C$. Any model i with a larger cost than C should therefore be omitted from the problem.

Design requirements extensions

We will here discuss examples of extra design requirements that may be imposed for a box set, such as particular models, particular themes or assigned difficulty levels.

If the company wants a box set to be themed, such as a car theme, they can, for example, require this in two ways: First, as a constraint that a particular set of models belong to the box. Second, that a particular number of pieces of some of the parts are included. For a car themed box set, this can be achieved by selecting a collection of car models must be in the box. Mathematically, we add the constraints $y_i = 1$ for $i \in \mathcal{L} \subset \{1, 2, \dots, M\}$, where $\mathcal{L}$ denotes the car models. Alternatively, a constraint that at least four pieces of the part "wheel" must be part of the box set would most likely allow for some car models. If j denotes a "wheel" part, we could then add the constraint $x_j \geq 4$.

Say the company wants the box set to include at least a given number of models for each of a given set of difficulty levels. Let the set $\mathcal{D}_l \subset \{1, 2, \dots, M\}$ denote the collection of models of difficulty level l. To ensure that a box set contains at least D_l models of difficulty level l, we can introduce the constraint $\sum_{i\in\mathcal{D}_l} y_i \geq D_l$.

Implementation example

We finish this section by showing an example of how the box set optimisation can be implemented. The optimisation problem is formulated with the Julia package JuMP [1], and then solved with Gurobi [2].

For the following example, the matrix P is generated randomly. It contains $M = 1000$ models with between 4 and $N = 17$ different parts. Each model consists of between 6 and 66 pieces. The part costs per piece are all set to the same number, $c_j = 1$ for $j = 1, 2 \ldots, N$, and we look for box sets with a total cost no more than $C = 70$. This can easily be formulated in Julia, and an example code is included in Listing 1. The optimisation problem takes less than two minutes to complete on a 32 core machine, and produces a box set that can generate 554 models of the 1000 models, using 70 pieces.

Acknowledgements

The Organising Committee of ESGI125 would like to 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. Many thanks also to the Cyprus University of Technology that provided venue, organisational and funding support, as well as to all our other sponsors and supporters, and particularly KPMG Cyprus (major sponsor).

References

[1] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. arXiv:1508.01982 [math.OC], 2015.

[2] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2016.

Listing 1: Simple Julia code that can formulate and run optimisation problem with the Gurobi solver.

```
using JuMP, Gurobi # The required Julia packages

# Assume we are given a matrix P of size M \times N

c = ones(N) # The cost array
C = 70      # The maximum cost

# Require at least D_l models of size >= 42
models_l = (1:size(P,1))[sum(P,2) .>= 42]
D_l = 4

# Set up optimisation model problem
m = Model(solver=GurobiSolver())

@variables m begin
y[i=1:M], (Bin, start=0) # Models, binary variables
x[j=1:N], (Int, start=0) # Parts, integer variables
end

@objective(m, Max, sum(y)) # Maximize number of models

@constraints m begin
dot(c,x) <= C
[i=1:models,j=1:parts], x[j] >= P[i,j]*y[i]
sum(y[i] for i in models_l) >= D_l
end

# The optimisation problem is solved by the solve(m) call
status = solve(m)

# The box set and models that can be created
boxpieces = getvalue(x)
boxmodels = find(getvalue(y))
```