**StrEmbed-5-3 functional description and issues**

*Notes on version 3 and plan/suggestions for version 4 and future developments; all actions carried over from StrEmbed-5-2*

Hugh Rice, June 2020

Overview: the main two elements that were to be added to version 3 of the software are as follows:

1. User selection of nodes in the lattice view by clicking the mouse generates new nodes – and therefore a new assembly – *via* lexicographic ranking/unranking. This consists of ordering possible configurations of parts according to lexicographic position. For example, in a ten-part assembly, the set of all possible five-part subassemblies begins with {1,2,3,4,5} and ends with {6,7,8,9,10}, where the integers represent the parts. This is important functionality because it allows calculation of lexicographic position (and therefore position in the Hasse diagram in the user interface) without realisation of all possible combinations, which can be extremely large (and therefore computationally difficult) even for assemblies of a few tens of parts.
2. Performing assembly/tree reconciliation, *i.e.* calculating the optimal number of node/edge deletions and additions (*i.e.* part and part-assembly relationships) necessary to transform one assembly to another. This functionality is intended as the first step in a larger set of similarity metrics (with other researchers in the project) to determine whether multiple assemblies can be represented in a single lattice, and if so, how, as efficiently as possible.

Both these elements have been incorporated into the software. Below is a table of other issues and suggestions for future versions of the software. Many are intended to optimise or speed up the software, rather than add functionality, but it is noted that the main speed bottleneck in the current version is related to plotting/visualising the Hasse diagram *via* the module Matplotlib, for which some suggestions for improvement are given below.

|  |  |
| --- | --- |
| *Description* | *Status and notes* |
| ***Minor issues/inefficiencies*** | |
| Compress “remove node” and “aggregate” operations into single method, and with “disaggregate” and “add node”, difference between what is returned by each pair of methods being data that are retained for future reference. | Not done as not significant |
| Width of border around images in geometry view should be increased as it’s not always clear which items are selected: selections are visualised by a change in colour of border. | Not done as not significant |
| Warn user if single-part assemblies are present when file loaded; provide option to remove. Required as single-part assemblies not allowed in Boolean lattice. | Done |
| Allow creation of new assemblies and manual linking of items to images. | First part done, second part not done as not significant |
| Reduce repetition of code in assembly operations, particularly between right-click and menu-initiated methods and in item filtering. | Partly done but could be expanded; however, wouldn’t affect speed of software much |
| Allow hovering with mouse over item to display item data in geometry and lattice views. | Not done as not significant |
| Create persistent image library to avoid reloading images for geometry view every time item is checked in parts view. | Not done as not significant |
| ***Major suggestions for version 4 and beyond*** | |
| Remove current module used for graph operations (Treelib) and replace with Networkx, which is more powerful and suitable for our purpose, specifically directed graph functionality. This will allow single object of custom class to be defined for each item. Discussed and agreed with TH. | Done |
| Create assembly management tools to allow user to track changes *via* (for example) “undo” and “redo” functionality. | Partly done: user can reconcile two assemblies |
| Allow visualisation of assembly in Hasse diagram, *i.e.* embedded in lattice. | Done *via* Hasse-like diagram |
| Create visual connection between different assemblies as join semilattices. | Partly done |
| Allow basic bill reconciliation tools. To be discussed with AMcK. | Done |
| Integrate some visualisation of machine learning results. To be discussed with TH and TC. | Not done as premature; to be discussed for version 4 |
| Integrate some adaptive system capability into what is presented to user in interface to incorporate TC’s work. To be discussed with TH and TC. We currently plan to identify a small number of specific ideas to realise. | As above |
| In lattice view, visualisations are via Networkx and Matplotlib FigureCanvas. Either (a) streamline with Matplotlib Picker/Artist functionality for cleaner code, or (b) replace entirely with (e.g.) Qtgraph, which is much faster, while retaining Networkx for assembly operations in background. This may also reduce need to redraw plot entirely each time an operation is performed and allow single items to be redrawn instead, for example. | Not done as Picker functionality found to be unsuitable; using Qtgraph in version 4 remains a possibility, to be discussed |
| Migrate software entirely from Wxpython to alternative GUI package Pyside2, which is open-source; FreeCAD is also written in Pyside2 and migration may allow better integration if we intend to create images via FreeCAD on the fly. | Not done as would be significant amount of work, but remains a possibility for future |
| Create images from items via FreeCAD on the fly. | Integration with FreeCAD, either by external script control of FreeCAD, or by use of FreeCAD GUI module in StrEmbed, is ongoing |