Horizon 2020

Societies of Symbiotic Robot-Plant Bio-Hybrids as Social Architectural Artifacts

Deliverable D3.3

flora robotica **as social garden**

Date of preparation: 2019/03/31	**Revision: 1**
Start date of project: 2015/04/01	**Duration: 48 months**
Project coordinator: UzL	**Classification: public**
Partners:	
lead: **CITA**	*contribution:* **UzL, UNIGRAZ, AMU, ITU, CYB**
Project website:	http://florarobotica.eu/

H2020-FETPROACT-2014

DELIVERABLE SUMMARY SHEET

Grant agreement number:	640959
Project acronym:	*flora robotica*
Title:	Societies of Symbiotic Robot-Plant Bio-Hybrids as Social Architectural Artifacts

Deliverable N^o:	Deliverable D3.3
Due date:	M48
Delivery date:	2019/03/31

Name:	*flora robotica* **as social garden**
Description:	This deliverable provides an overview of progress towards fulfilling the objectives of the final task in WP3 - developing *flora robotica* as an Internet-connected Social Garden. A physical Social Garden has been constructed and we report upon the infrastructures developed to support various levels of communication and interaction within it, and across Social Gardens via a cloud-based API. We report upon the development of virtual environments to support the investigation of braid morphologies, symbiont morphologies, and controllers using interactive artificial evolution setups. We also report upon the architectural design of the physical setup and a hypothetical Social Garden proposition for an urban site. Finally we report upon targeted growth dynamics of the Social Garden, with a particular focus on artificial growth of the scaffold and self-repair behaviours. We conclude that the Social Garden objectives originally set out in the project proposal have been fulfilled.

Partners owning:	**CITA**
Partners contributed:	**UzL, UNIGRAZ, AMU, ITU, CYB**
Made available to:	public

Contents

1 Introduction

This deliverable is the last in a series of three that report on the development of the architectural aspects of the bio-hybrid robotic symbiont. In D3.1 we reported on the development of specific design rules together with appropriate representation and simulation methods required to support the design of our bio-hybrid architecture. In D3.2 we reported on the development of specific construction logics based on braided systems, and presented initial architectural propositions exploring the growing of spaces and the growing of building components. The principle task in this work package targets the integration of the core technologies developed over the course of the project as a physical site - a 'Social Garden' - that showcases bio-hybrid functionality.

Content overview

The Social Garden represents the integration of sub-systems to produce a functioning bio-hybrid robotic symbiont, operating towards architectural objectives.

In Section 2 of this deliverable we report on communication infrastructures established to support interactions within a Social Garden, and across Social Gardens via a cloud-based API.

In Section 3 we report upon the development of virtual environments for exploring bio-hybrid morphologies, controllers and braided strucutres using interactive artificial evolution.

In Section 4, we report on the design of the physical Social Garden site. This physical site presents a number of constraints on design, therefore we also develop a speculative Social Garden proposition for an urban context. We present this proposition together with a brief summary of the design workflow we have established to develop it.

In Section 5, we report upon the observable dynamics exhibited by the benchmark demonstrator and within specific sub-systems.

In Section 6, we provide a summary conclusion of the achievements and contributions presented within this deliverable.

2 Communication infrastructures

This section reports upon the communication infrastructures developed to support local and remote interactions within and between Social Garden sites.

2.1 Within a social garden

We start with a report on infrastructures for supporting real-time, in-situ communication between the Social Garden sub-systems.

2.1.1. Plants and distributed robotic nodes for shaping

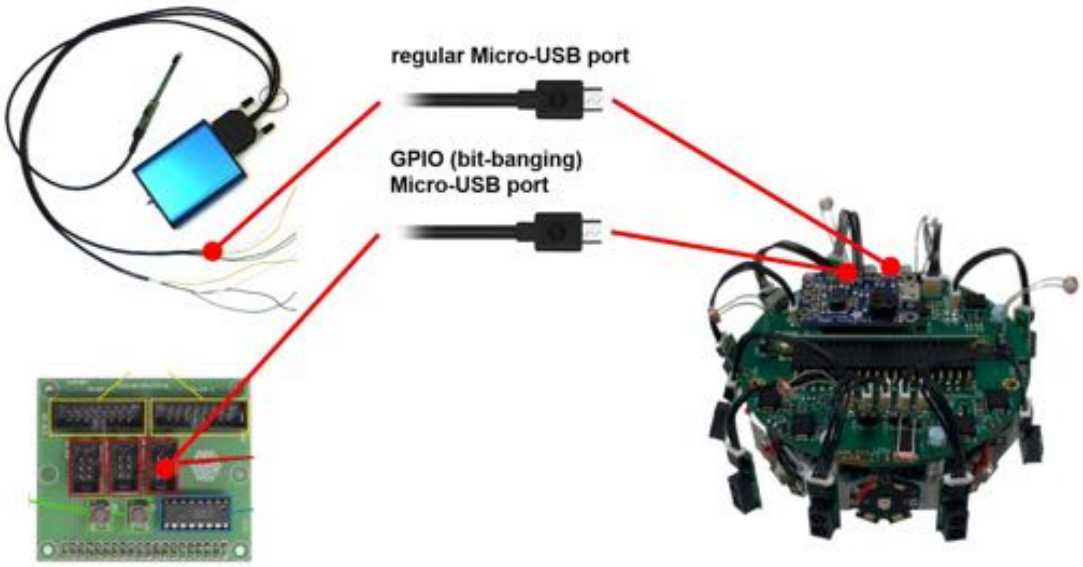


Figure 1: Main robotic node for plant shaping (right) has Micro USB ports for extension, which can support regular USB port connection to for instance the phytosensing module (left, top) and can also support GPIO (bit-banging) USB port connection to Raspberry-Pi based devices such as the Raspinet for braided structures (left, bottom).

The main robotic node for plant shaping (described in Sec. 5 of *D1.4 Evaluation of the robotic symbiont*) is an extensible hardware platform. In addition to powering and controlling extension modules for supplementary plant shaping by stimuli or auxin inhibitors (see Sec. 5.2.2 of D1.4), the main nodes have open-ended opportunities for further extension. The main nodes include three Micro USB ports, which support I^2C, GPIO communication. The ports support connection to the MU phytosensing module produced by Cybertronica (see Sec. 4 of *D1.4 Evaluation of the robotic symbiont*). In the context of plant shaping tasks, the phytosensing module can perform detection of damage in plant tissues, or detect touch-based human interaction with the plants. The ports also support serial communication with Raspberry Pi based devices, such as the Raspinet devices for artificial growth of braided structures by the Vascular Morphogenesis Controller (VMC), described below. Likewise, the ports can support devices that connect to and control the distributed actuator nodes (see Sec. 3.1 of *D1.4 Evaluation of the robotic symbiont*)

for further shaping of braided structures. In this way the extensible hardware approach allows the integration of control for shaping artificial growth with that for natural growth. The ports allow for further extension modules. This feature, combined with the open-source bio-hybrid experiment protocol (see Sec. 5 of *D1.4 Evaluation of the robotic symbiont*) and the Social Garden cloud API (see below), allows for open-ended extensions of the platform beyond the end of the project.

2.1.2. Raspinet on braids

Design improvements. Raspinet[1] has seen a major improvement in the communication between its single modules, greatly reducing experiment times and CPU usage, as well as increasing the flexibility of the system. Initial investigations, software updates and physical experiments regarding the fusion of branches were performed, allowing for the formation of more complex (interconnected) networks than binary trees. A new Python-based visualization of the graphs formed by the Raspberry Pi network was realized.

Communication between modules: serial bit-banging with pigpio. Whereas previously, to communicate with up to five other Raspberry Pis via the GPIO pins, we had used a fuzzy method involving software PWM, we now switched to *exact* serial bit-banging using pigpio[2]. With PWM communication, it could take up to 15 minutes for a reading to fully converge to the value sent by a neighbor, particularly if the change was big. This was due to averaging over large numbers of voltage readings on the respective input pins to read back the PWM frequency set by the sender. The pigpio daemon's background handling of the serial communication is far superior to this averaging procedure, that had to be running in parallel threads for each of the five input pins of a module, enabling a drastic reduction in CPU usage by roughly 90% down to about 10%. The reduction in self-written code needed was not quite as drastic, yet still substantial: pigpio offers to comfortably use various communication protocols (e.g., serial, I^2C, SPI, etc.) on any of the general purpose IO pins of a Raspberry Pi via software control. Generally, this allows far greater flexibility in the types of algorithms that can be implemented with the system: Now, a single message can be more complex, if required.

More relevant to our case, since information transfer is now immediate, all inertia in the response of a module to changed circumstances in its neighbors stems purely from the parameterization of VMC (the *Vascular Morphogenesis Controller*, see for example *Deliverable D2.4*, Sec. 7 for the most recent update) and not from drawbacks in the communication protocol. Technically, each message between the modules also contains the unique ID (encoded as an 8-bit integer) of the sender, but this only plays a role in the network visualization (see below) and has no influence on the decisions of the VMC algorithm.

Implementing fusion of branches in hardware and software. In the current implementation of Raspinet, running the VMC algorithm, agents send and receive a single 16-bit integer representing the amount of *resource* or *successin* (i.e., the success-hormone generated at the leaves of the graph), depending on whether the sender is, respectively, a parent or a child of the receiver. The flow of successin toward the root node increases the width of the edges (i.e., what we call *vessel thickness* here, in analogy to plants' dynamic vascular system [7]). The resource is generated at the root node (in analogy to the aggregated input from a root system of a plant

[1] Raspberry Pi based electronics mounted on hand-braided, branching modules as initially described in [5] and *Deliverable D2.3*, Sec. 3.4. More recent experiments are presented in this Deliverable in section 5.2.

[2] `http://abyz.me.uk/rpi/pigpio/`

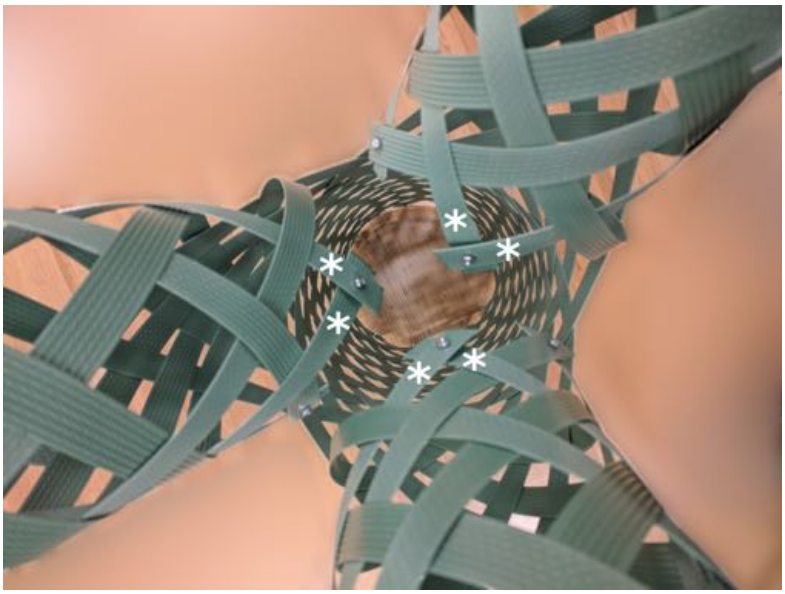


Figure 2: Bottom view of braided module allowing three branches of distinct Raspinet modules to fuse. Each of the three branches of this module inserts into an arm of a regular Y-branching Raspinet module. The trunk of this module serves as the host to insert the child module shared by the three fused parent modules. The white asterisks mark the two extra filaments braided into each arm at the junction to achieve compatibility with the host branches.

to its above-ground shoot via the main stem) and distributed among its children in proportion to the relative thickness of the edges (or vessels) connecting them.

However, communicating only one value leaves a child node unaware of the vessel thickness to its parent. This information is stored in the parent node, and based on it, the node divides the resource among its children. In the tree graphs formed by VMC *without branch fusions*, a node needs not know the vessel thickness towards his parent, since each module only has a single parent to which all of its successin is transferred. With fusions between branches, questions arise how to distribute successin among multiple parents. The options we considered are: Distribute incipient successin 1) equally; 2) depending on the received resource; 3) depending on the vessel thickness toward the parent.

From these, the only way to keep fused cross-connections stable in any given (static) environment is to equally distribute the successin between its parents (option 1). The other two options impose positive feedbacks that amplify small differences such that only one of the two parent connections "survives" in the long term. This is because the successin itself directly (option 3) or indirectly (option 2) leads to an increase of these variables.

To physically enable the braided structures to merge branches, we build dedicated braided modules that don't carry electronics themselves, but only relay the cables connecting the parents to their common child. A single module can have up to three parent modules, requiring a three-to-one arm module (Fig. 2). Fusing two branches into one, a simple Y-branch module (with inverted screw orientation to avoid jamming) can be used.

Graph visualization. Every active Raspinet module (i.e., every non-leaf node of the VMC graph) publishes its state to the local WiFi network using ZeroMQ publisher sockets.[3] The state comprises a list of numbers describing the internal variables (resource, successin and vessel thickness), the values received from and sent to neighboring modules as well as all sensor-readings. Using the new mode of communication (see above), each module now attaches its unique identifier number to the messages sent to its neighbors. In turn, all neighbor IDs received by a single module are labelled with the type of neighbor (parent 1-3 or child 1-2) and published to the WiFi as well.

A desktop computer (on the same network) continuously receives and aggregates the messages sent by all modules. At regular intervals, a *snapshot* of the whole VMC system is taken: A CSV file containing the information from all nodes of the network is stored. This file can be directly interpreted to reconstruct and plot the graph of the system using the neighbor IDs of each node, whereas before we had to provide a manually updated adjacency matrix of the network whenever there was a change to the graph (i.e., an addition or removal of nodes or edges).

The graph reconstruction and visualization is now handled by a new Python-based suite harnessing the power of the networkx library[4] for representing the graph data. Previously, we have made use of the visualization capabilities of our VMC simulation platform (written in the Processing language[5]) that has been used in many other contexts (e.g., see Zahadat et al. [17], Hofstadler et al. [5], Zahadat et al. [18] and *Deliverable D2.3*, Sec. 3). But for the sake of flexibility and compatibility with the rest of the Raspinet universe (written in Python), we found it a worthy investment to build a custom tailored solution. Crucially, the current implementation can also handle fusions. Additionally, with the VMC system parsed into a networkx graph, we are free to export it to various specialized third party network analysis and/or visualization software. To this end, for visualization of a VMC system at any given point in time, we rely on network layouts generated by Graphviz[6], that we adapt according to our needs. See Figs. 56, 57, 58 and 61 for inlays of the VMC graphs of the physical structures shown.

Interacting with the robotic plant shaping nodes. With hardware based on Raspberry Pi as well, the robotic nodes for plant shaping[7] can connect straightforwardly to the Raspinet modules. An interface for iteration with the robotic nodes for plant shaping has been realized (see Fig. 1). The current generation of robots has already been designed with that in mind.

The plant shaping robots can physically connect to any of the available plugs on a Raspinet module, either as a parent or a child. Attached as a root, they use one of the three parent-plugs on the Raspinet *root-board*. That means that they cannot attach to junctions where three arms fuse into one, as the three parent modules occupy all plugs. When attached as a leaf, they connect to the single child-plug of one of the two branches of a Raspinet module. Should a growth event happen at such a branch, the plant shaping node is replugged to the newly added Raspinet module as a root (parent).

[3] http://zeromq.org
[4] https://networkx.github.io
[5] https://processing.org
[6] https://www.graphviz.org
[7] The first generation of the robotic plant shaping nodes is detailed in *Deliverable D1.2*, Sec. 2.2, while the latest incarnation is presented in *D1.4*, Sec. 5 and *D2.4*, Sec. 2.

2.1.3. Plant-Plant, Human-Biohybrid-Plant, and Robot-Plant interactions

Social garden assumes interactions, thus, one of tasks in development of the phytosensor system was related to 'interactions'. Further we distinguish between

1. human – biohybrid-plant interactions;
2. plant – plant interactions;
3. robot – plant interactions.

Human – biohybrid-plant interactions. This is most common type of interactions, it is conducted via temperature changes, gas concentration (primarily CO2), in capacitive or mechanical ways (e.g., by touch or vibrations) from human side. Plants react via RGB light, mechanical actuators, messaging system (e.g., twitter) or voice interface. Reactions and responses are programmed by using the DA scripts (see description of the phytosensor system). Examples are shown in Figs. 6 and 8.

Plant – plant interactions. Extension of 'human–plants' interactions are 'plant-plant' interactions. They use RGB light, sound, and mechanical actuators to generate stimulus signals. Examples are shown in Figs. 4 and 5. These experiments were multiple times demonstrated in different mass media, see, for example, Fig. 7 with Danish DR2 TV.

Figure 3: Interactions between two plants and a robot arm.

Robot – plant interactions. This type represents to some extent a simplified version of previous schemes, however it has a high 'demonstrative power'. Mechanical actuator, for example, 5 DoF robot arm with USB-PWM controller, is connected to the phytosensor (via USB hub). Movement is controlled by DA scripts, which in turn are activated by signals from plants.

This approach enables controlling the actuator directly by the plant (as reactions on electrophysiologcal or physiological biosignals). Behavioural patterns are either dynamically generated by the script or are switched between multiple pre-programmed patterns. More complex approaches are possible. Example of the setup is shown in Fig. 3.

Thus, the developed interaction approaches and their implementation in the phytosensor system enable a wide range of socially-interactive scenarios as, for example, the Social Garden, artistic installations, interactions with auditoria, exhibitions, and others.

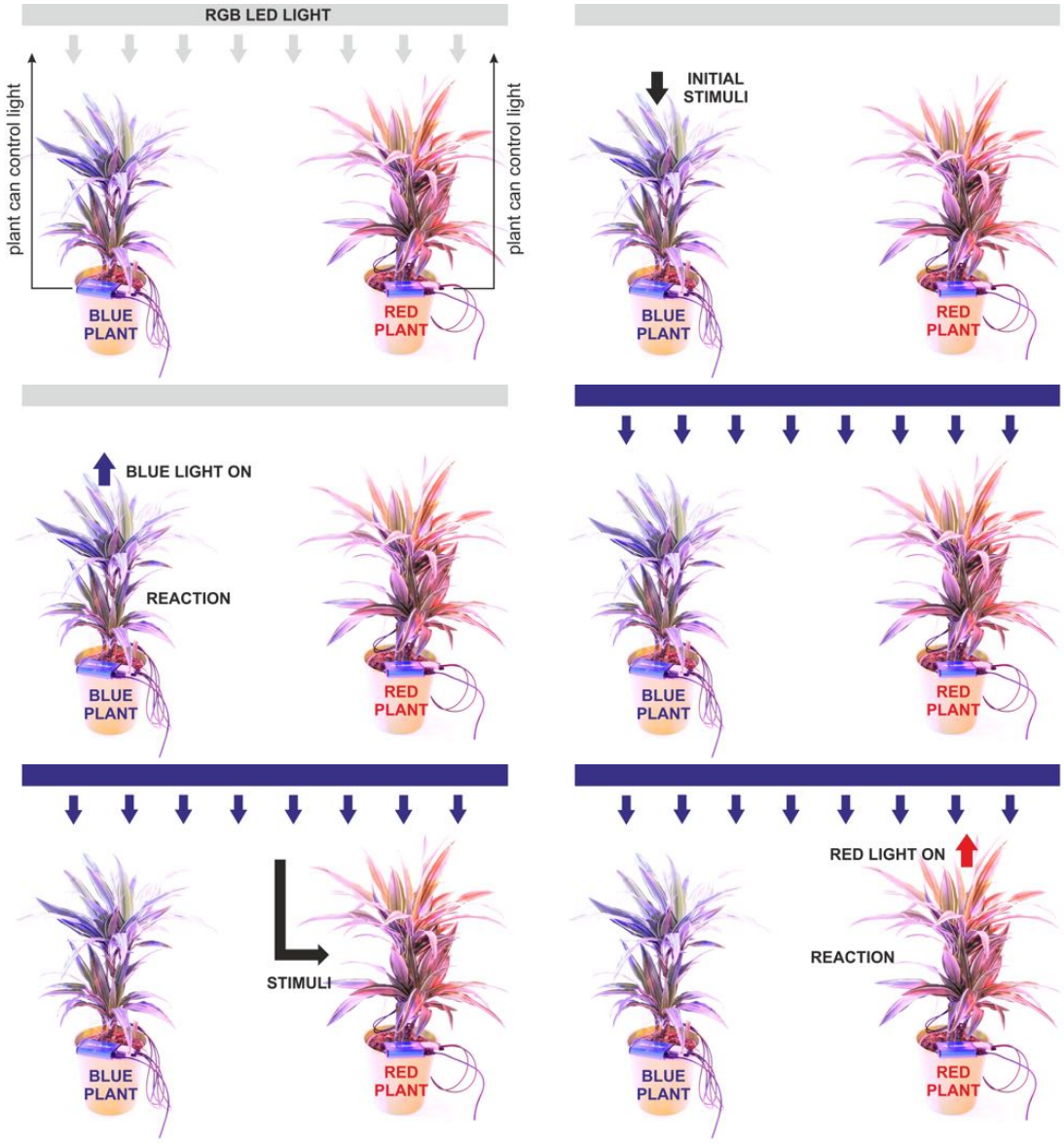


Figure 4: Interactions between two plants via RGB light and sound signals (part 1).

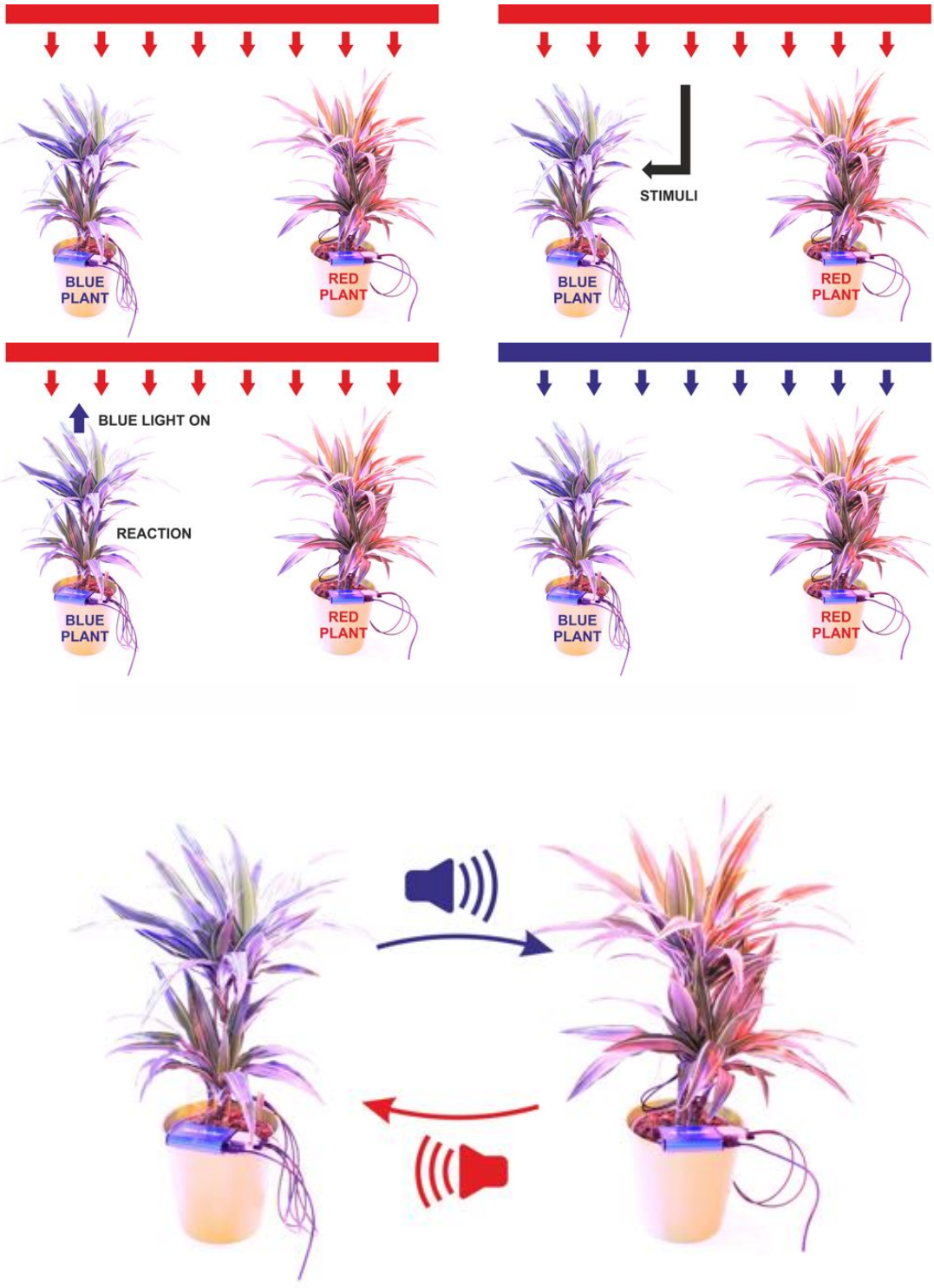
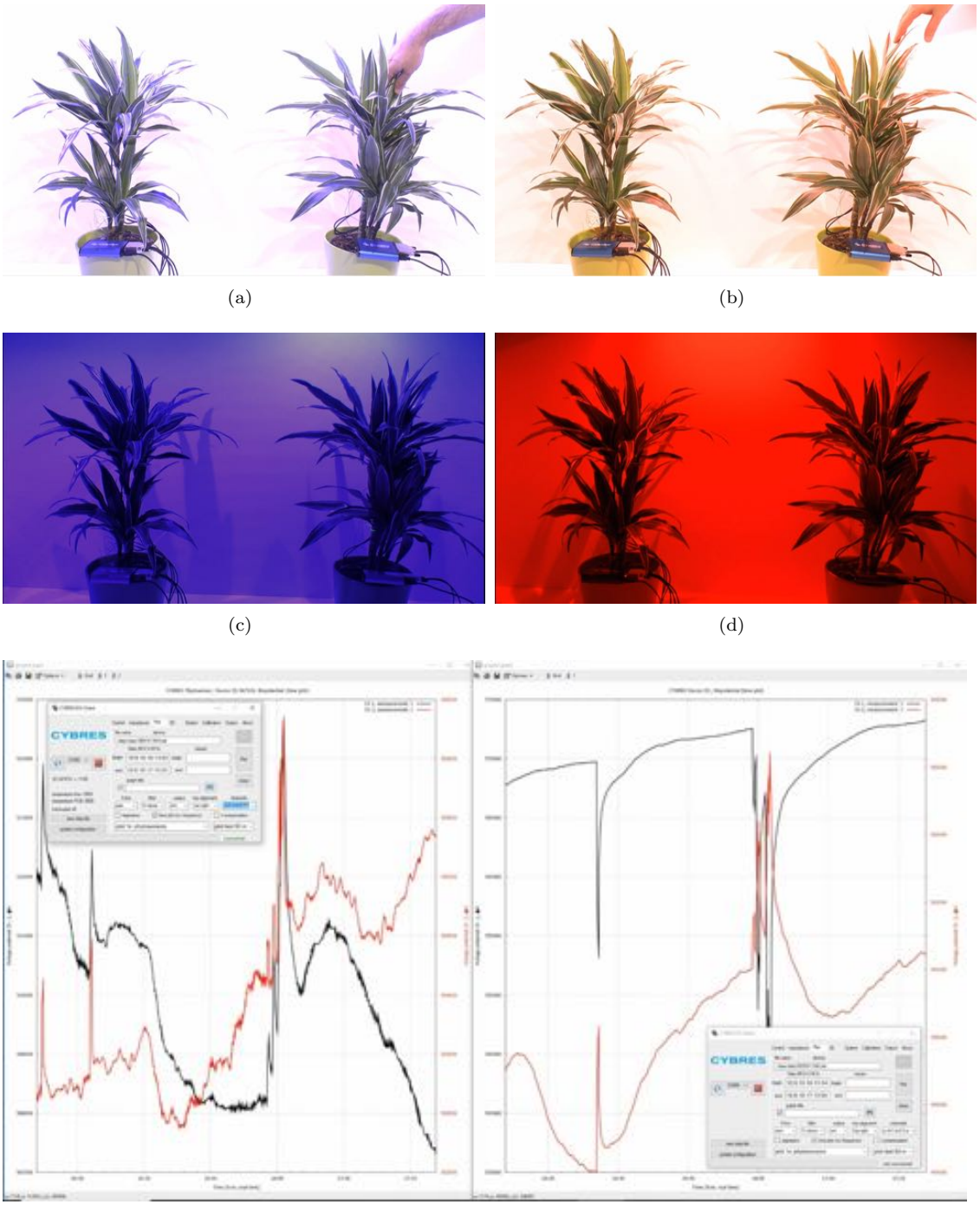


Figure 5: Interactions between two plants via RGB light and sound signals (part 2).



(a) (b)

(c) (d)

(e)

Figure 6: (a)-(d) Interaction between one plant and human user (via touch) that triggers in turn interactions between plans (via light); (e) biopotential signalling between plants in this experiment.

Figure 7: Demonstration of the 'two plants experiment' to Danish TV.

Figure 8: Demonstration plant-to-voice interface and interactions with human user.

2.2 Across social gardens

2.2.1. Social Garden cloud API

To experiment with setups across different *flora robotica* sites, we setup the *social garden API*, a client site tool for sending and retrieving data from a social garden cloud.

In more detail, the API, has been setup to send REST CRUD (Create, Read, Update and Delete) operations to the database via a web service. This API facilitates connecting devices and sensors to a central server. The python implemented API provides the following functionalities:

```
getNewData(StreamName) - Get new data points, updates every 0.5 seconds.
streamExists(StreamName) - Check if stream exists in database.
getAllData(StreamName) - Get all data from stream.
getLastData(StreamName) - Get last data from stream.
insertStream(StreamName) Insert stream in database.
deleteStream(StreamName) - Delete stream from database
```

An example of connecting to a specific stream and reading out attributes is shown below:

```
import socialgardenapi
print('Starting listener')
while True:
    resp = socialgardenapi.getNewData("myStream")
    print(''TimeStamp: "+resp['TimeStamp'])
    print(''StreamName: "+resp['StreamName'])
```

The data and their origins can be visualized through a simple web interface (see Fig. 9). In this example, data from a phytosensor by Cybertronica (CYB) is visualized and can be openly accessed online.

2.2.2. Open-access data sets and open-source software

In addition to the possibility of live-steaming sensor and other data through the Social Garden cloud API, the project provides an open-access database of several data sets produced during the project, through the Zenodo[8] platform, as well as open-source repositories through GitHub,[9] as specified in *D4.1 Data management plan (open research data pilot)*.

1. **Self-Organized Construction with Continuous Building Material**
   (a) **Data types.** Experiment videos; experiment images; paper figures.
   (b) **License.** Creative Commons Attribution 4.0 International.
   (c) **Citation.** Heinrich, M. K., Wahby, M., Divband Soorati, M., Hofstadler, D. N., Zahadat, P., Ayres, P., ... Hamann, H. (2016). Self-Organized Construction with Continuous Building Material: Higher Flexibility based on Braided Structures. Zenodo. http://doi.org/10.5281/zenodo.58524
   (d) **Abstract.** Self-organized construction with continuous, structured building material, as opposed to modular units, offers new challenges to the robot-based construction process and lends the opportunity for increased flexibility in constructed artifact properties, such as shape and deformation. As an example investigation, we look at continuous filaments organized into braided structures, within the context of bio-hybrids

[8] http://www.zenodo.org, Zenodo is developed by CERN under the EU FP7 project OpenAIREplus (grant agreement no. 283595).
[9] https://github.com/

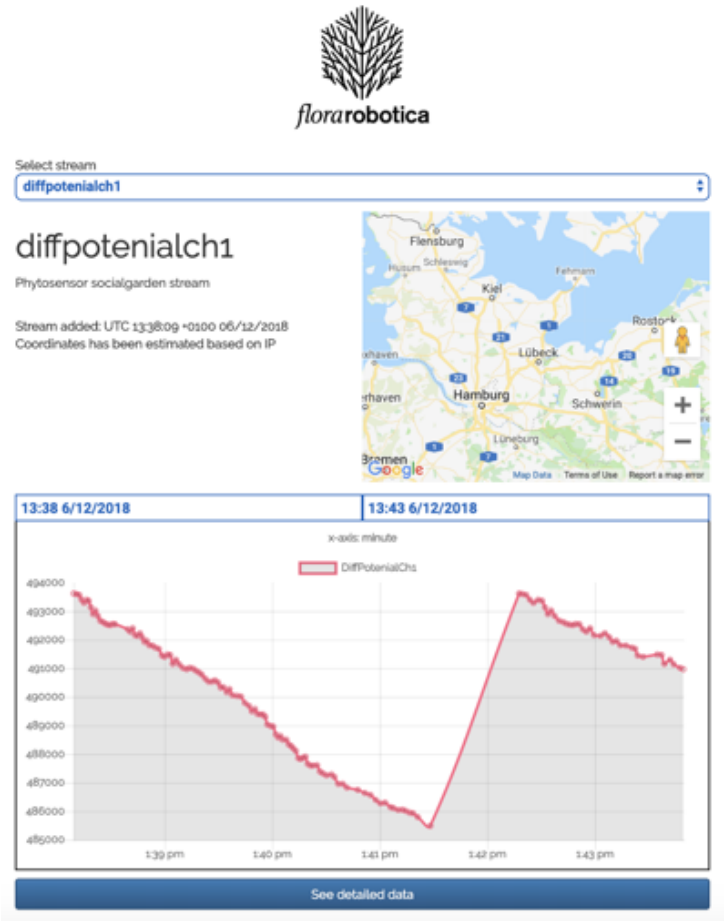


Figure 9: Social Garden API

constructing architectural artifacts. We report the result of an early swarm robot experiment. The robots successfully constructed a braid in a self-organized process. The construction process can be extended by using different materials and by embedding sensors during the self-organized construction directly into the braided structure. In future work, we plan to apply dedicated braiding robot hardware and to construct sophisticated 3-d structures with local variability in patterns of filament interlacing.

2. **Robot Self-Assembly as Adaptive Growth Process**

   (a) **Data types.** Experiment videos; experiment images; paper figures.

   (b) **License.** Creative Commons Attribution 4.0 International.

   (c) **Citation.** Divband Soorati, M., & Hamann, H. (2016). Robot Self-Assembly as Adaptive Growth Process: Collective Selection of Seed Position and Self-Organizing Tree-Structures.
   Zenodo. http://doi.org/10.5281/zenodo.58703

   (d) **Abstract.** Autonomous self-assembly allows to create structures and scaffolds on demand and automatically. The desired structure may be predetermined or alternatively

it is the result of an artificial growth process that adapts to environmental features and to the intermediate structure itself. In a self-organizing and decentralized control approach the robots interact only locally and form the structure collectively. Designing a complete approach that allows the robot group to collectively decide on where to start the self-assembly, that adapts at runtime to environmental conditions, and that guarantees the structural stability is challenging and does not yet exist. We present an approach to self-assembly inspired by diffusion-limited aggregation that generates an adaptive structure reacting to environmental conditions in an artificial growth process. During a preparatory stage the robots collectively decide where to start the self-assembly also depending on environmental conditions. In the actual self-assembly stage, the robots create tree-like structures that grow towards light. We report the results of robot self-assembly experiments with 50 Kilobots. Our results demonstrate how an adaptive growth process can be implemented in robots. We explain how our approach will be extended to a 3-d growth process and how robot self-assembly as an open-ended adaptive growth process opens up a multiplicity of future opportunities.

3. **Braid Tiling Experiments**

   (a) **Data types.** Simulation experiment results.

   (b) **License.** Creative Commons Attribution 4.0 International.

   (c) **Citation.** Zwierzycki, Mateusz Jan, Vestartas, Petras, Heinrich, Mary Katherine, & Ayres, Phil. (2017, March 22). Braid Tiling Experiments. Zenodo. http://doi.org/10.5281/zenodo.437462

   (d) **Abstract.** This set of images presents a set of experiments which were done to test the capabilities of the developed braiding pattern generation methods. Each figure shows a different kind of tiling problem, from regular to completely random patterns.

4. **Microscale braid geometry relaxation**

   (a) **Data types.** Simulation experiment results.

   (b) **License.** Creative Commons Attribution 4.0 International.

   (c) **Citation.** Vestartas, Petras, Zwierzycki, Mateusz Jan, Heinrich, Mary Katherine, & Ayres, Phil. (2017). Microscale braid geometry relaxation. Zenodo. http://doi.org/10.5281/zenodo.438631

   (d) **Abstract.** Yarn yarn collision of high resolution meshes, derived from graphs.

5. **Autonomously Shaping Natural Climbing Plants**

   (a) **Data types.** Experiment videos; experiment sensor data logs.

   (b) **License.** Creative Commons Attribution 4.0 International.

   (c) **Citation.** Wahby, Mostafa, Heinrich, Mary Katherine, Hofstadler, Daniel Nicolas, Neufeld, Ewald, Kuksin, Igor, Zahadat, Payam, Hamann, Heiko. (2018). Autonomously Shaping Natural Climbing Plants: A Bio-hybrid Approach. Zenodo. http://doi.org/10.5281/zenodo.1172160

   (d) **Abstract.** Plant growth is a self-organized process incorporating distributed sensing, internal communication and morphology dynamics. We develop a distributed mechatronic system that autonomously interacts with natural climbing plants, steering their behaviours to grow user-defined shapes and patterns. Investigating this

bio-hybrid system paves the way towards the development of living adaptive structures and grown building components. In this new application domain, challenges include sensing, actuation and the combination of engineering methods and natural plants in the experimental set-up. By triggering behavioural responses in the plants through light spectra stimuli, we use static mechatronic nodes to grow climbing plants in a user-defined pattern at a two-dimensional plane. The experiments show successful growth over periods up to eight weeks. Results of the stimuli-guided experiments are substantially different from the control experiments. Key limitations are the number of repetitions performed and the scale of the systems tested. Recommended future research would investigate the use of similar bio-hybrids to connect construction elements and grow shapes of larger size.

6. **Survey of participant experience in workshop for testing software setup**
   (a) **Data types.** Survey responses data; plots.
   (b) **License.** Creative Commons Attribution 4.0 International.
   (c) **Citation.** Heinrich, Mary Katherine, Zahadat, Payam, Harding, John, Nicholas, Paul, & Ayres, Phil. (2018). Survey of participant experience in workshop for testing IGP software setup: supplemental data set for SimAUD 2018 (Version v1) [Data set]. Zenodo. http://doi.org/10.5281/zenodo.1196027
   (d) **Abstract.** These survey results are regarding the experience of participants in a workshop testing our software setup for Interactive Evolution of self-organizing growth, including an implementation of the Vascular Morphogenesis Controller, and the Interactive Evolution software Biomorpher. The full-time one-week workshop was held as part of the normal coursework of the Master's degree program CITAstudio: Computation in Architecture, in the Institute of Architecture and Technology, at KADK, The Royal Danish Academy, School of Architecture, Copenhagen, Denmark. It was part of the first semester of the 2017-2018 school year. Workshop participants were current Master's students in the CITAstudio program.

7. **"IGP-for-Grasshopper"**
   (a) **Data types.** Python repository; Grasshopper3D user objects.
   (b) **License.** MIT License.
   (c) **Citation.** https://github.com/florarobotica/IGP-for-Grasshopper.
   (d) **Abstract.** Grasshopper implementation of Interactive Evolution setup, using 3D VMC and Biomorpher.

8. **"polymesh-braid"**
   (a) **Data types.** Visual Basic repository.
   (b) **Citation.** https://github.com/florarobotica/polymesh-braid.
   (c) **Abstract.** The methods and classes provided in this library can be used to generate a braiding-oriented directed graph, which can be then translated into mesh geometry using the MeshWrap Class. Finally you can use this geometry to obtain the braiding pattern from the Weaver Class.

9. **"Kangaroo2-Braid-Simulation"**
   (a) **Data types.** C# repository.

(b) **Citation.** https://github.com/florarobotica/Kangaroo2-Braid-Simulation.

(c) **Abstract.** Custom workflow to simulate yarn-yarn collision. C# (Visual Studio) for use in Kangaroo2 within Grasshopper3D.

3 Interactive evolution for social gardens

In this section we report on the development of interactive evolution in the exploration of bio-hybrid morphologies and controllers.

3.1 Interactive evolution of structures grown by VMC

The software setup supports architectural design of self-organizing behaviors via Interactive Evolution. It extends the user interface of interactive evolution (in this case, in Biomorpher) to support the user's understanding of non-deterministic self-organizing behaviors—and therefore support their ability to make informed judgments during artificial selection. We illustrate the description of the setup with an example design task of obstacle avoidance.

3.1.1. Interactive evolution approach

Biomorpher platform for Interactive Evolution. A plug-in for that supports interactive evolution (IE) in Grasshopper3D, Biomorpher[10] uses a version of a Cluster-Oriented Genetic Algorithm (COGA) [8]. COGAs encompass a two-step process, first using a diverse search algorithm to rapidly explore the design space, and second, adaptively filtering the population in order to present the user with concise, digestible clusters of solutions. At each generation, Biomorpher allows the user to choose either evolution by artificial selection, or optimization according to performance objectives. These can be combined into multi-mode optimization, by alternating the two modes between generations. COGAs normally filter and cluster solutions according to fitness value, helping the user to choose high performing regions for refinement [2]. When Biomorpher optimizes according to performance objectives, it uses a version of this approach.

Here we employ exclusively Biomorpher's artificial selection mode. When Biomorpher evolves with artificial selection, it uses k-means clustering [6] to group solutions according to parameter state similarity. For each of 12 defined clusters, a representative solution is presented to the user.[11] The user selects one or more of these representatives to define parents for the next generation of evolution. When using artificial selection, each cluster representative is accompanied by an indicator of performance for supplied criteria, allowing the user to incorporate quantitative feedback into their judgment and selection.

In order to support the specific use-case here, a modification is implemented to Biomorpher, enabling the visualization of user-defined mesh colors in the selection preview windows.

User Interface for Self-organization in IE. In Biomorpher, the parameter states are supplied separately from the mesh geometry that will display in the respective preview window for user selection. As such, the setup can control what the user will see during selection.

The setup provides two IE user interface functions: 1) simultaneous viewing of multiple possible results, and 2) visualization of the environment.

In the user interfaces of prominent IE projects for creative production [11, 3], the user is endeavoring to evolve a static image or 3D shape. In these cases, the visualization of evolved solutions is straightforward—each preview window shows the image or shape created by the respective parameter state. In our case, however, visualization is less straightforward. Because non-deterministic behaviors will generate variability and unpredictability in resultant structures, any given parameter state might produce a range of results. Every time a non-deterministic controller runs, it will produce a different result. The degree to which the results can vary will

[10] https://github.com/johnharding/Biomorpher

[11] See an illustrative video demonstration: https://www.youtube.com/watch?v=EM6uoXW7Yeo

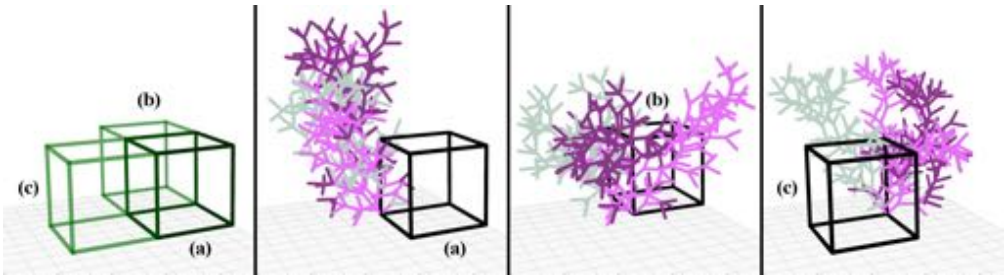


Figure 10: Visualization of the environment to which the controller is currently responding during growth. Three example environments are shown here, for the task of obstacle avoidance. Each environment (a,b, and c) contains a box-shaped obstacle in a different location (all three obstacles shown, left). At the right and center, the VMC growth can be seen responding to each respective environment. Also seen here is simultaneous viewing of multiple possible growth results (VMC structures), differentiated by color.

depend on the behavioral properties of that individual controller. In the setup, we therefore display multiple possible results simultaneously in each IE preview window (see Figures 12, 10). Each displayed result is given a separate color so the user can differentiate. The number of results shown at a time is user-defined (here, three solutions are shown). This interface feature performs several functions. First, it promotes user understanding of the inherent variability present in

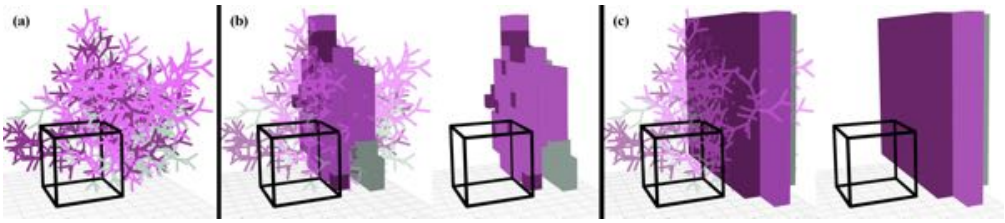


Figure 11: (a) VMC structure without a second layer of control for material aggregation, (b,c) the same VMC output, with its respective structure when incorporating simple rule-based control for material aggregation, with either (b) control for placing bricks, or (c) control for placing a solid wall.

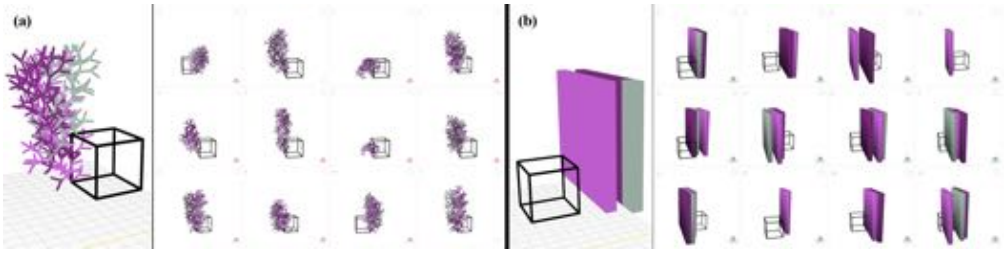


Figure 12: (a) Direct representations of graph-based structures grown by VMC; a generation of such growths in the Biomorpher IE preview windows. (b) Solid wall structures defined by hybridizing the VMC with rule-based control for material aggregation; a generation of such structures in Biomorpher.

non-deterministic self-organization. Second, showing the user a range of possible results helps them avoid the selection of a parameter state that solves the task by coincidence, as opposed to solving it reliably through a feature of behavior. Third, seeing a range of results allows the user to evolve not only the behavior itself, but the degree of variability in that behavior's results (see Fig. 12).

Simultaneous viewing of multiple results in the IE preview window is accomplished by consecutive independent simulations of controller behavior, for each parameter state queried by the EA.

When designing a behavior that grows a structure in response to the environment, a visualization of the respective environment is evidently useful. In our case, showing the environment is intended to support the user in making informed judgments about the behavior or response, rather than only the artifact. Importantly, the inclusion of an environment visualization in the IE preview window also allows the environment to be randomized at each generation without confusing the user. In our setup, at each query of a parameter state, the environment data used in simulation of the controller's behavior is randomized from a predefined list or external simulation (see Figs. 10 and 12). Randomizing the environment at each generation helps the user avoid getting stuck in false optimums by evolving towards a controller that can coincidentally solve the task only in one environment, as opposed to solving it reliably in any environment of the relevant type.

The environment in which the structure grows in simulation is represented in by an *Environment Data File*, containing a 3D matrix of the values that would be obtained by sensors in a matching reality setup. The wireframe box shown here (see Figs. 10, 11, and 12) does not represent the process by which the controller responds to environment in simulation, but rather is a visualization tool for the user to understand the current environment.

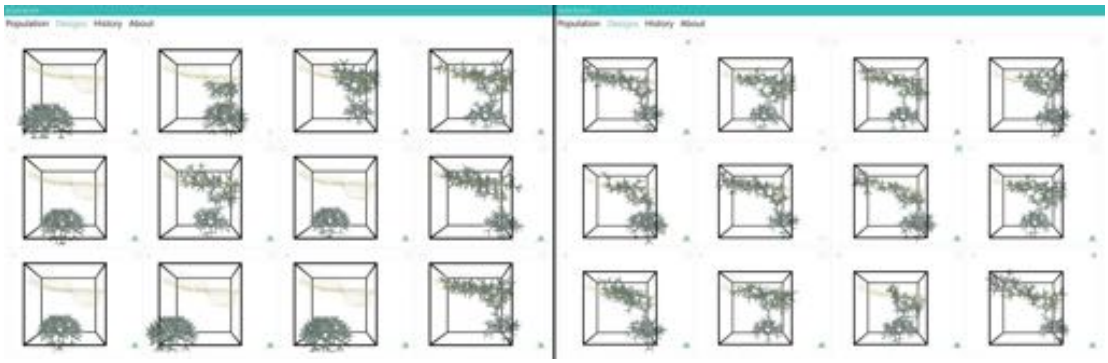
From control logic to physical structure. Instead of direct relationships between the physical and the encoding—where each edge and vertex of the VMC graph has a one-to-one relationship with a physical element in the structure, which restricts the possible material structures to tree-shapes—we extend to flexibility in physical structure by decoupling the control of material aggregation from the control logic that discovers environmentally advantageous locations for growth. A second layer of control for material aggregation allows for the distribution of heterogeneous tasks, and therefore the creation of hybrid control. Such hybrid control can provide the flexibility typically sought in early phase architectural design.

When being hybridized with an existing self-organizing controller (in this case, the VMC) control for material aggregation can itself also be self-organizing, or can instead be simply rule-based. Here we illustrate with rule-based control for placement of basic building elements: bricks or walls (see Fig. 11). In this scenario the VMC can be used to sense and avoid the environmental obstacle *in situ*, thereby solving the specified task reliably through features of its behavior, while the second layer of control—for material aggregation—allows the architect to have flexibility in the design of an artifact, rather than being restricted to tree-shaped structures.

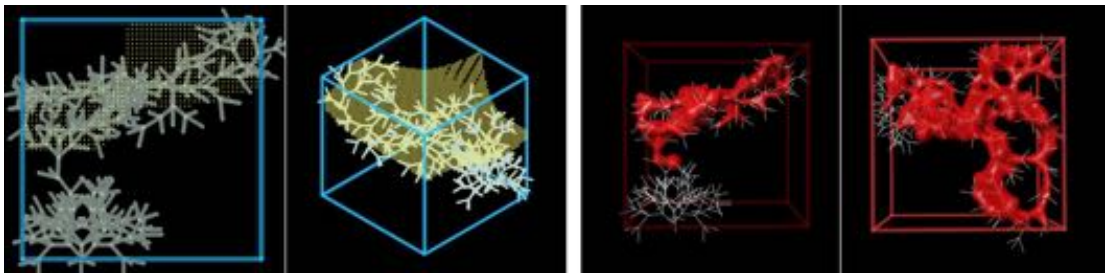
After applying hybridized control, the final resulting material aggregations can be viewed and evolved in the IE setup (see Fig. 12), making use of the previously described features of simultaneous results and environment visualization.

3.1.2. User tests

The software setup was provided to architectural designers and engineers, during a one-week workshop including tutorials and group project work. Each group used the software setup to



(a) Screenshots of the interactive evolution process, using the VMC and our IE setup. Left, an early, random generation. Right, a generation where the behavior results shown seem to indicate convergence on an area of parameter states that reliably solve the design task at hand.



(b) Designers' visualizations, output from the final controllers designed in the setup. Left, the VMC locating the area of light conditions where the group's chosen design task requires the placement of a barrier. Right, the structure resulting from the group's hybridization of the VMC with a second self-organizing controller for design-driven amorphous material aggregation.

Figure 13: Project results from the group focused on response to sunlight.

design a controller for a distinct design task. After the workshop's end, the participants received a survey about their experiences with the software setup and relevant concepts.

User-made Design Projects. After tutorials on concepts and software, each group of participants chose a scope for their project. They chose either a building component to shape (e.g., doorway, column) or an environmental condition to respond to (e.g., thermal, occupant circu-

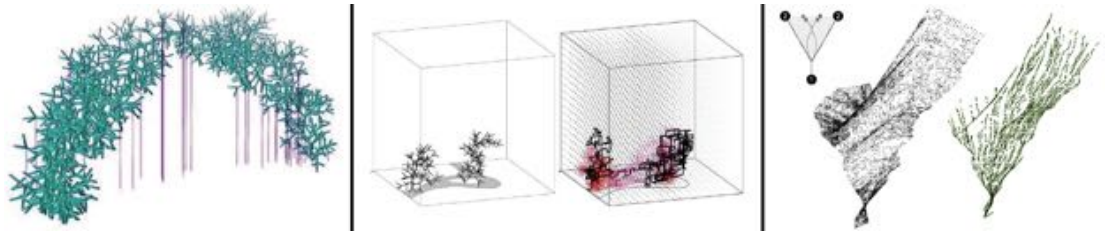


Figure 14: Designers' visualizations of their final hybrid controller outputs for (left) a bridge, (center) shielding a sidewalk from adverse wind, and (right) forming a basin to collect rainwater at an advantageous position.

lation). Within the selected scope, each group defined a design task and worked to create a controller that could reliably solve that task in simulation, in at least three different environments.

Survey of User Experience. The workshop participants gave survey responses anonymously. Survey responses were collected via Google Forms.[12] At the start of the survey, participants gave permissions for use and publication, and verified that they participated in the workshop and had not previously taken the survey. The platform discourages duplicate responses by requiring an email sign-in (which is not visible to the surveyor).

Although workshop participants gave permission for survey results to be published before taking the survey, the participants were unaware of the specific intended context and purpose of publishing, prior to taking the survey. Workshop participants had no contact with the process of survey preparation or analysis of its results. There were 26 workshop participants. Participants were architects or architectural designers.

Participants were asked about 1) their prior experience, 2) their understanding of topics before and after the workshop, 3) the helpfulness of specific software aspects for their understanding and their project work, and 4) their likelihood to use specific software aspects in the future.

In addition to looking at the full surveyed group, we compare experience sub-groups. Participants select relevant tasks that they have previously completed, from a provided list. They are placed in the *Less Experience* sub-group if they select one or no tasks, and in the *More Experience* sub-group if they select two or more.

Workshop Results. Two groups chose building components to shape—a bridge, a staircase—and three groups chose an environmental condition to respond to—sunlight, wind, rainfall.

Four out of five groups were able to successfully use the setup to design a non-deterministic self-organizing behavior to reliably construct artifacts that solved their chosen design task (see Figs. 13 and 14). All four of those groups used the full set of software aspects provided, including interactive evolution and the addition of their own layer of control for material aggregation, hybridizing with the VMC. As well, the second layer of control designed by each group incorporated some aspect of self-organizing behavior in the decision-making process for material aggregation. The sunlight group (see Fig. 13) was furthermore able to evolve the parameters of the 3D VMC to reliably find a specific desired feature in multiple example environments, then using the second layer of their own control to aggregate material in a design-driven way. The bridge group (Fig. 14, left) used decentralized decision-making to find topology features in the VMC graph structure that were advantageous as initiation points for their material aggregation. The wind group (Fig. 14, center) responded to feedback from the sensed environment not only with the VMC's behavior, but in the behavior of their own swarm agent controller for material aggregation. The water collection group (Fig. 14, right) used their material aggregation control to find and build off of useful shape features in the volume of the VMC's growth, rather than simply refilling the zone defined by the VMC with a different type of structure. This group was able to usefully distinguish between tasks for the VMC and their own control rules, such that both are necessary for their hybridized controller to be successful at its task.

The staircase group did not incorporate separate control for material aggregation, but after the workshop instead took the approach of extending the VMC's behavior to solve their design task. In this group's approach, once an instance of the VMC finishes its growth—according to its parameters and the values sensed in the environment—their extension globally selects a leaf of the resultant graph structure to become the root location for a new VMC instance. The leaf

[12] https://www.google.com/forms/about/

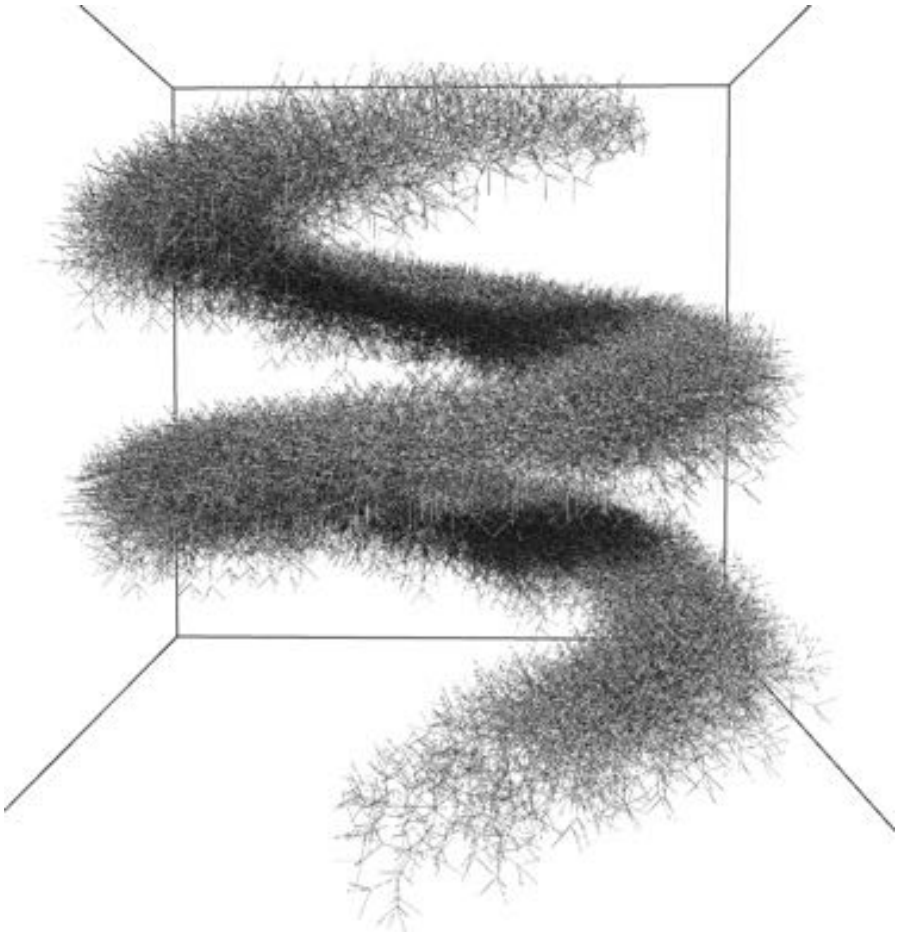
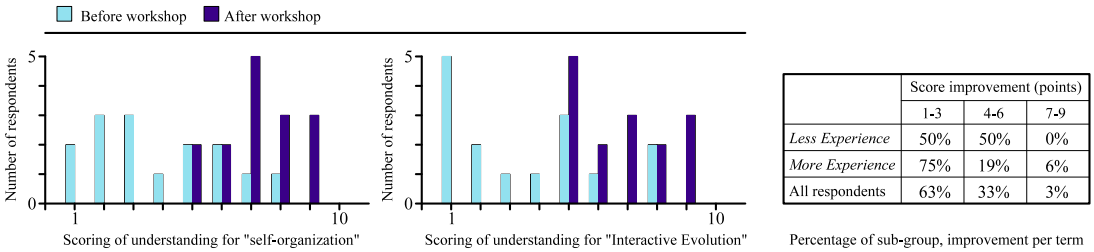


Figure 15: Designers' visualization of their staircase, output from their extended VMC implementation.

that becomes a new root is selected according to globally assessed features of overall height and proximity to desired staircase shape. Although the group completes this process of new root selection through global control, a decentralized decision-making process could potentially be implemented to achieve a similar result, thereby substantially broadening the types of structures able to be grown with VMC.

The survey responses indicate that no workshop participant had previously designed a self-organizing controller using interactive evolution, so the project results and survey results give strong evidence that the setup helped the participants to understand this design approach enough to use it to solve their chosen design tasks.

Survey responses. Close to two-thirds of workshop participants submitted survey responses (16 of 26, or 61.5%), with at least two respondents per group. One respondent indicated workshop absence; their responses were removed. One respondent indicated that they did not understand two questions, so those two responses were removed. All responses were submitted within 18 days of workshop end.



	Score improvement (points)		
	1-3	4-6	7-9
Less Experience	50%	50%	0%
More Experience	75%	19%	6%
All respondents	63%	33%	3%

Percentage of sub-group, improvement per term

Figure 16: Participants' scoring of their understanding of the topics "self-organization" and "Interactive Evolution" respectively, comparing scores before and after the workshop.

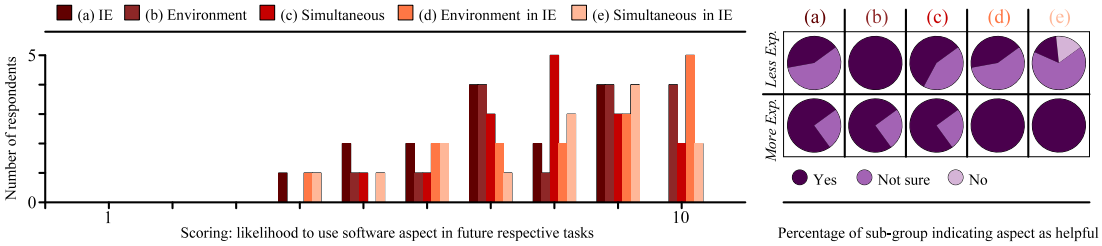


Figure 17: (left) Participants' scoring of their likelihood to use certain aspects of the software setup again, if they were to design a non-deterministic self-organizing behavior, and (right) participants' indications of the helpfulness of those same software aspects.

Responses regarding understanding (see Fig. 16) give evidence that the software setup helped participants of both experience levels improve their understanding of related topics. Those with less prior experience improved their understanding more than others, and understanding of "Interactive Evolution" improved slightly more than understanding of "self-organization."

Responses regarding the usefulness of certain software aspects (see Fig. 17) give evidence that: 1) interactive Evolution helped participants to understand and design a non-deterministic self-organizing behavior (see Fig. 17, right, a); 2) visualization of the environment and simultaneous viewing of multiple results helped them to understand and design such behaviors (see Fig. 17, right, b and c); and 3) the setup's features of environment visualization and simultaneous results inside the artificial selection preview windows of the IE setup helped them to evolve behaviors to solve their chosen tasks (see Fig. 17, d and e).

The survey responses indicate that no workshop participant had previously designed a self-organizing controller using interactive evolution, so the project results and survey results give strong evidence that the setup helped the participants to understand this design approach enough to use it to solve their chosen design tasks.

3.2 Interactive evolution of braided structures

In Interactive Evolutionary Computation (IEC) humans takes the role of the fitness function by iteratively selecting from a set of candidate solutions. IEC has been used in various domains where personal preferences can help guide the evolution and search the problem space of a given domain. These projects have been especially useful for creating artefacts such as music, pictures, and 3D shapes. Here we explore the use of IEC to allow casual users to invent their own braided 3D structures.

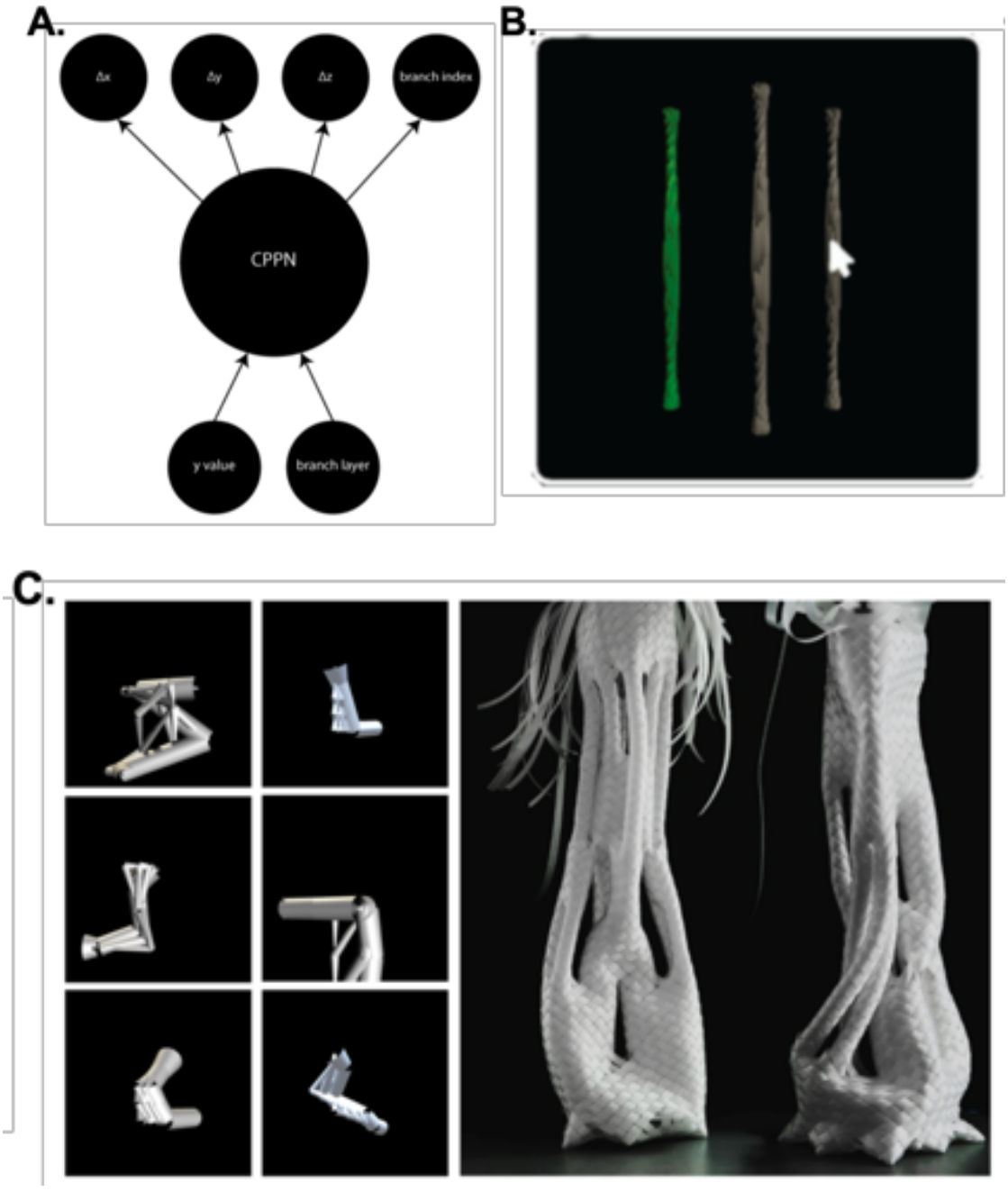


Figure 18: Interactively evolved 3D braid structures. (A) CPPN representation. (B) IEC user interface. (C) Evolve structures (left) and examples of hand-designed and fabricated structures (right).

The 3D artefacts are encoded by a special type of neural network, a *compositional pattern producing network* (CPPNs; [12]) and evolved by the NEAT algorithm [13]. Initial braids are modelled as a cylinder consisting of n connected nodes with different y coordinates. The braid-encoding CPPN takes these y coordinates as input to decide where to branch the cylinder or what offsets to add to each node's x and z position. In more detail, the CPPN (Fig. 18A) receives two inputs, the y value of the node and the branching layer of the node. The CPPN has four outputs: delta values of x, y, z that are added to the node vector value, while the branching output determines whether a node should branch if its value is above a certain threshold. If the node branches, a new series of child nodes with a different branching layer are added to the node and then given as input to the CPPN.

The IEC-interface is shown in Fig. 18b. At each iteration users can select between three different braid designs and the next generation is then created by slightly mutating and combining the selected CPPNs.

Fig. 18C shows a comparison between some of the evolved braided structures from interactive braid evolution and handmade braid structures fabricated at CITA. While IEC allows users to evolve a diversity of interesting shapes, the handmade structures are currently still more complex. In the future, additional CPPN extension (e.g., an output that can merge seperated branches back together) and a collaborative IEC approach with many users could narrow this gap.

4 Architectural design of social gardens

In this section we report upon the setup design for bio-hybrid growth—which can also be considered a physically constructed Social Garden—and the design of a hypothetical Social Garden proposed for an external urban context.

4.1 Setup design for bio-hybrid growth

This bio-hybrid setup is for tests of shaping plant growth along scaffolds using stimuli (not scaffold shape), via the plant shaping robotic nodes described in Sec. 5 of *D1.4 Evaluation of the robotic symbiont*.

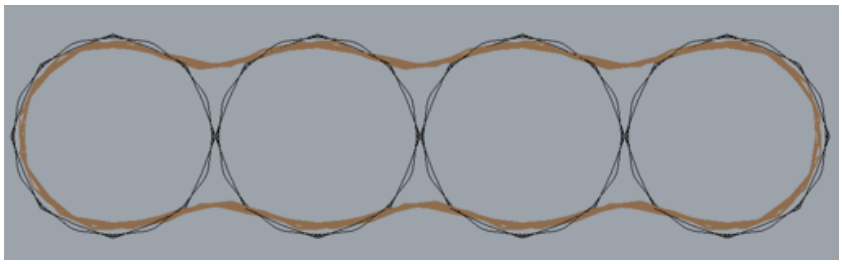


Figure 19: Top view of the design for a scaffold module for the bio-hybrid growth experiment setup. Module consists of four braid tubes (black lines) joined by a second layer of braid (brown lines). The individual braid tubes can be manufactured by a small set of modules of the reconfigurable braid machine (see Sec. 2 of *D1.4 Evaluation of the robotic symbiont*) and then joined together by a second layer, or with a larger set of braid machine modules, the full 4-tube panel could be produced at once.

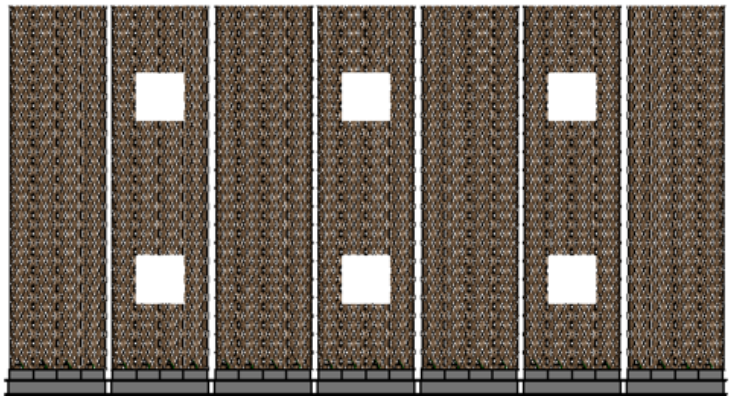


Figure 20: Front elevation view, non-perspective, of a full lab room setup with seven individual experiment modules at the stage of experiment initiation (i.e., before plant growth). Modules alternate between "Damage" and "Window" conditions. Modules with "Window" conditions have holes, as they are desired and should be maintained by the plants throughout the experiment. "Damage" conditions do not have holes at this stage, as the holes are undesired and made after the first round of plant growth, to then be repaired by the second round of plant growth.

In each panel, the plants and nodes will solve either a "Window" task (i.e., stay out of the desired window) or a "Damage" task (i.e., repair the undesired damage), see Fig. 20. In the "Window" task, the hole in the scaffold is present at experiment initiation and is maintained throughout. In the "Damage" task, the scaffold has no hole at experiment initiation, and the plants are steered to grow throughout the full scaffold; after the plants have grown over the area, a "Damage" hole is then created, including cutting the plants that have grown in that area; new plants are then steered to grow over the "Damage" hole to begin repairing it. Each panel is made of 4 wooden braid tubes made by machine, joined by an additional layer of hand-braided wood (see Fig. 19). With a larger set of braid machine modules, the full double-layer 4-tube construction could be braided by machine.

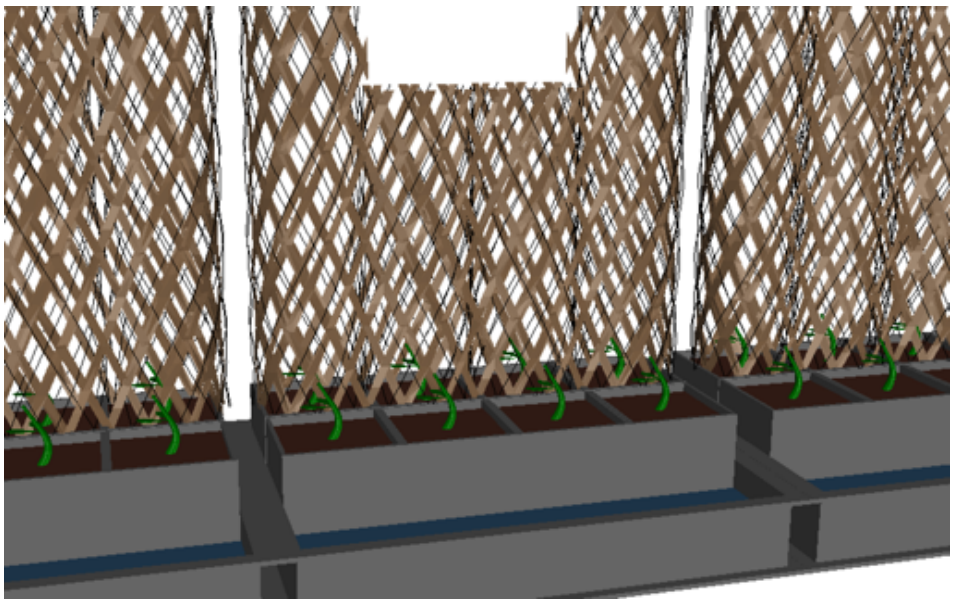


Figure 21: A mainly front-facing perspective view of the bio-hybrid experiment setup; close-up on the base condition of one "Window" module between two "Damage" modules that have not yet been damaged, as the experiment is at initiation stage. Here is shown that the "Window" condition being tested in the module is close to soil level, to help reduce overall experiment time.

To support experiment documentation with time-lapse photography in a lab room that is very narrow, the setup is relatively flat, against one wall of the room. Likewise for recording view angles, nothing is in front of or behind the experiment setup, with background elements in white color to reduce visual confusion. The undesired "Damage" holes and desired "Window" holes are kept identical in terms of surrounding scaffold condition, to demonstrate that stimuli steer the plants, not the shape of scaffold. Portions of the scaffold that are not "Windows" or "Damage" are kept uniform to again allow the demonstration of stimuli-based steering to not be disrupted by scaffold shape. When a hole is cut for "Damage,' the surrounding scaffold within the module remains self-supporting overall. Some test conditions (i.e., "Windows" or "Damage") are placed very close to the roots (see Figs. 21 and 22), supporting the possibility of relatively fast trial-and-error, as experiments may become disrupted by external lab conditions (e.g., power outage, watering malfunction, flash malfunction).

The modular sectioning approach to the scaffold allows experiments to be replicated, and repetitions to run simultaneously. Also through the sectioning approach, plants are easy to

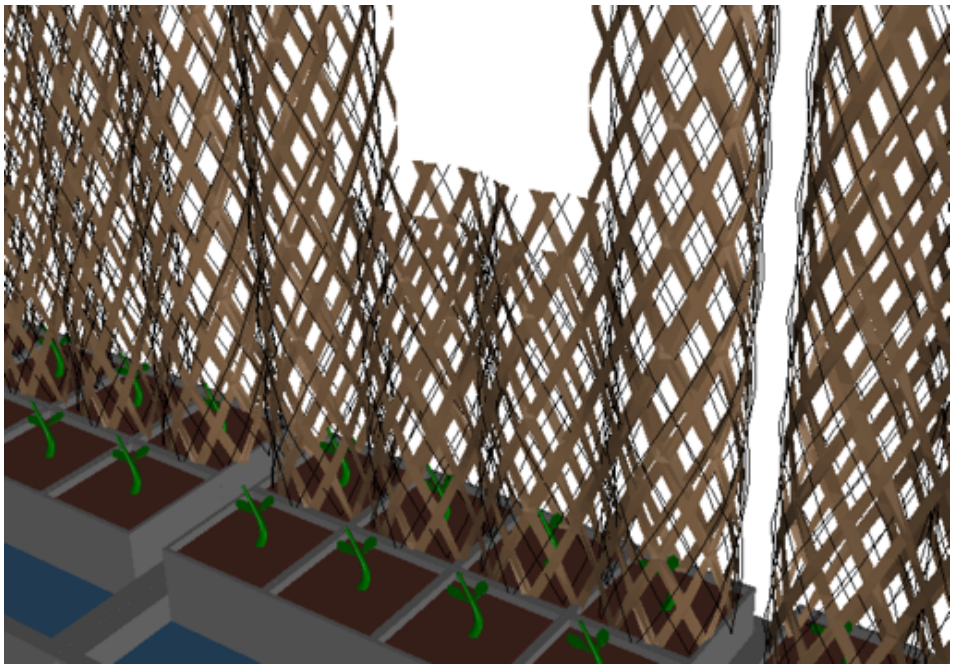


Figure 22: A mainly top-facing perspective view of the setup. Here the hollow condition of the braid tubes can be seen. The setup includes both a front row and back row of pots with soil and plant shoots. The back row of plants can attach to the front braid wall from behind, while the front row of plants can attach from the front. This gives the experiment the possibility to test at least eight plants, an increase from the at least four plants tested in prior experiments on a diagrid setup (see Sec. 5.1 of *D1.4 - Evaluation of the robotic symbiont*).

replace if they die, due for instance to bugs, watering issues, or other health problems outside of the conditions being tested in the experiments. The approach does not limit the ability to construct larger structures, as modular sections of scaffold can alternatively be used as base in a 'pre-fabrication' approach (see Fig. 23). It is also possible to reorient the individual sections, forming a 'corner' condition (see Fig. 24). They can be joined together with an additional braid layer, allowing cross-growth of plants between modules. In the setup here this could be enacted at heights above completed experiments. The scaffold is made of flat wooden strips, as flat strips hold each other in place when in a hollow tube braid, in a way that rods or thin circular-cross-section filaments cannot. Wood is used as a bio-degradable material that is known to be attractive to plants as a climbing substrate, and does not have unknown impacts on plants from features sometimes found in synthetic materials such as off-gassing.

This setup is for the task of plant shaping by stimuli, not by shape of scaffold. The possibilities of augmenting this function via shaping by scaffold are explicated in the following section.

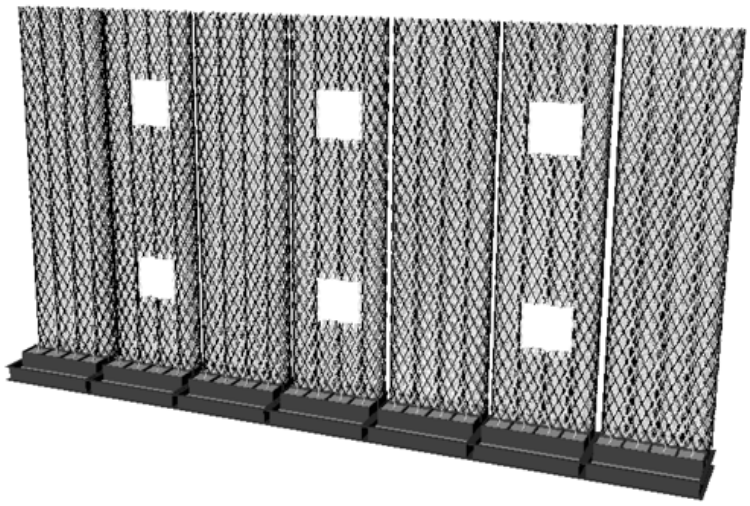


Figure 23: Perspective view of a full scaffold setup, showing the possibility of flat wall construction.

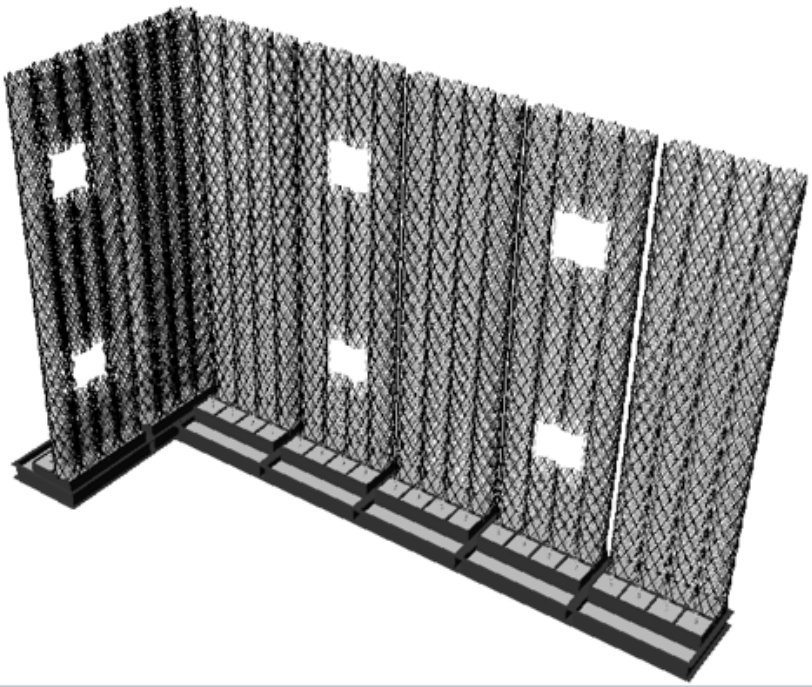


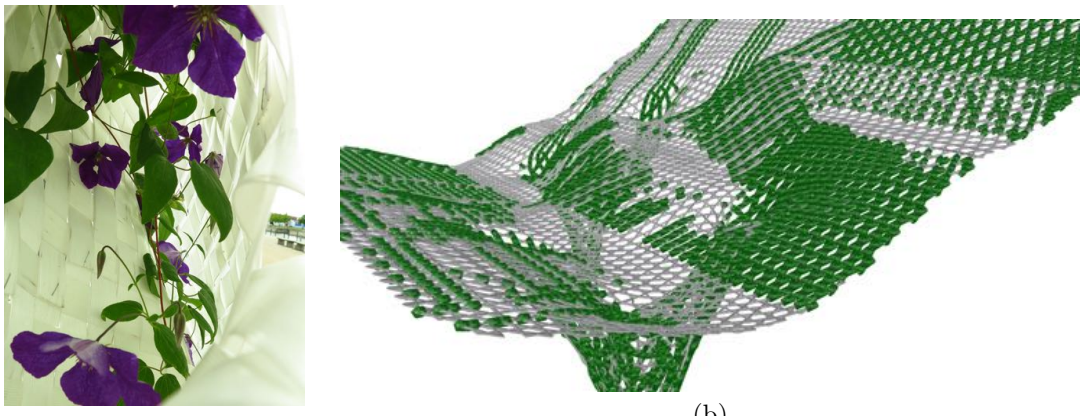
Figure 24: Perspective view showing the possibility of a corner construction, using the modular scaffold approach.

4.2 Design propositions for an urban site

This section presents a speculative architectural proposition for an urban site based on the application of key concepts, technologies and scientific insights from *flora robotica*. We also report on the computational design approaches developed to investigate and produce propositions, showing how we integrate relevant design approaches with site based environmental analysis, structural analysis and plant growth simulation. From an architectural design perspective, this is the core contribution of the work presented in this section.

4.2.1. Living weaves concept

The *flora robotica* project has established the use of braided scaffolds as an integral element used to support early stage plant growth (Fig. 25a), to have embedded robotic and adaptive capabilities, to act as a site for the location of robotic nodes and to mark out spatial architectural intentions in anticipation of future growth. The project has also demonstrated the use of distributed and decentralised robotics to steer plant growth, using stimuli and along scaffolds, towards intended shape.



(a) (b)

Figure 25: (a) Plants supported by the scaffold; (b) plants integrated into the structure to provide additional structural capacity.

Here, we speculate on the integration of these two project findings to grow plants towards shape that can provide future structural capacity (Fig. 25b). We propose that plant branches are treated as a directable filament that can be fully integrated into a braided structure, acting as tri-axial members of the braid to produce a sparse Kagome pattern [1]. The *Living Weaves* construction concept is shown in Fig. 26 in simulation (a), and, as a physical assembly (b, c) combining Fallopia vines manually woven through lightweight maple bi-axial scaffolds as growth progresses. Fig. 26c shows broad cover having developed from the dormant vine stock over a three-week period in controlled conditions. Growth rates were measured at approximately 40 mm/day. There is no observable change to woody tissues in the new growth yet, but this is to be expected later within the same growth season with Fallopia.

(a) (b) (c)

Figure 26: (a) Simulation of plant growth in the structure, replacing a structural member of the triaxial weave; (b) detail Fallopia growth in lab condition; (c) early stage growth on structure.

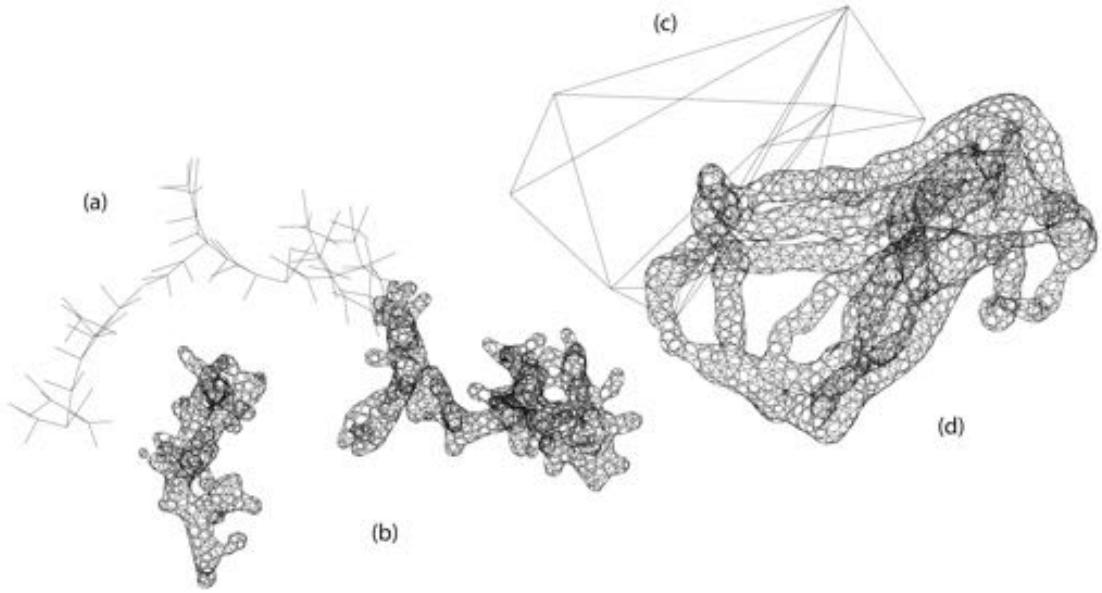


Figure 27: Graph and Line input to Kagome weave: (a) graph input; (b) weave pattern from graph. (c) line input; (d) weave pattern from lines.

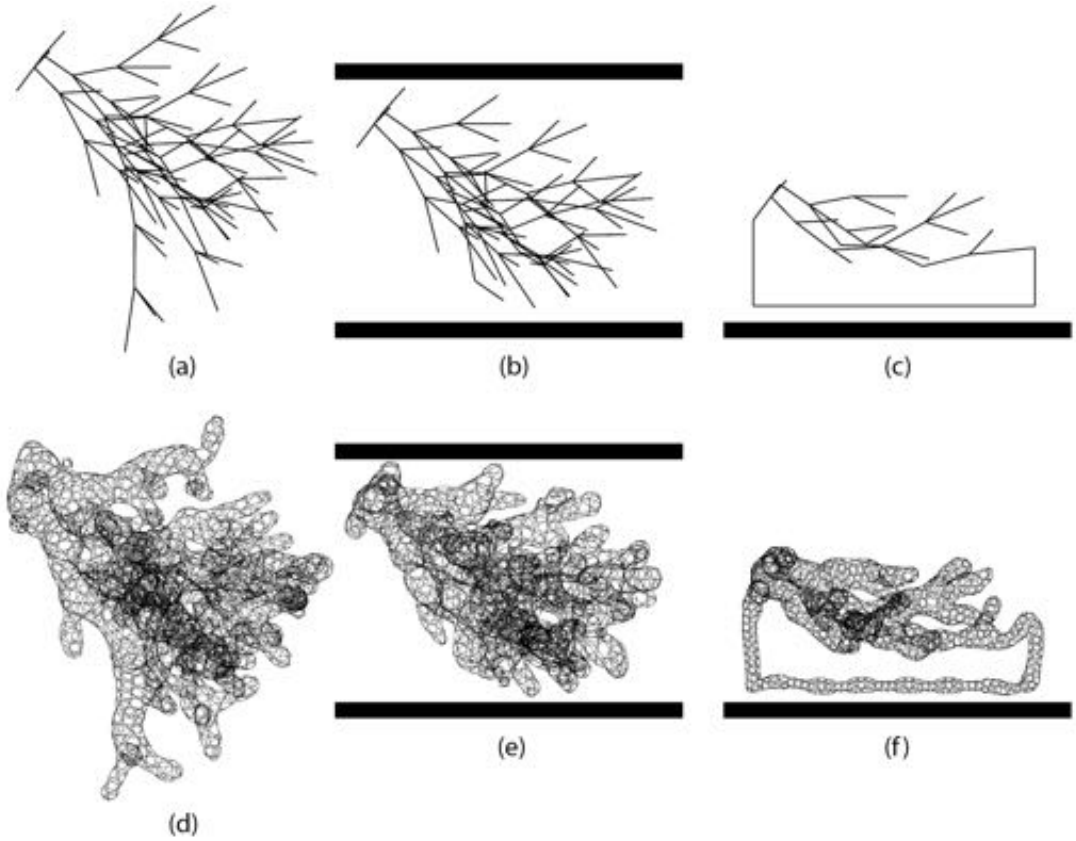


Figure 28: Context adaptation: (a) graph-based lines as weave pattern input; (b) pruning of graph-based input for site adaptation (walls); (c) top-down designer decision for line input; (d) weave from graph-based lines; (e) weave from pruned graph-lines; (f) weave from designer intervention.

4.2.2. Design approach

In order to develop the visualization and analysis approaches for working with the *Living Weaves* concept, it has been important to use lightweight computational simulation methods which can give a quick feedback for decision making and design adaptation. Additionally, it is crucial to have different methods for design speculation and multiple possibilities for design feedback to explore and exploit the inherent dynamics across the lifecycle of the bio-hybrid system. For the proposition presented here, the design workflow primarily revolves around the generation of the scaffold structure which follows Kagome principles [1].

The topology of the weave structure can be built from computationally generated bottom-up, artificially grown graphs (Fig. 27a), top-down drawn lines (Fig. 27c) or by data-informed voxel grids which we describe below. The graphs can be derived from the output of the Vascular Morphogenesis Controller (VMC), a distributed controller for growing artificial structures [16], or generated numerically using the networkx library [4] for python [15]. Line drawings can be based on architectural context (spatial features), top-down decisions of the designer or derived from structural analysis in later feedback scenarios. A specific architectural context may necessitate

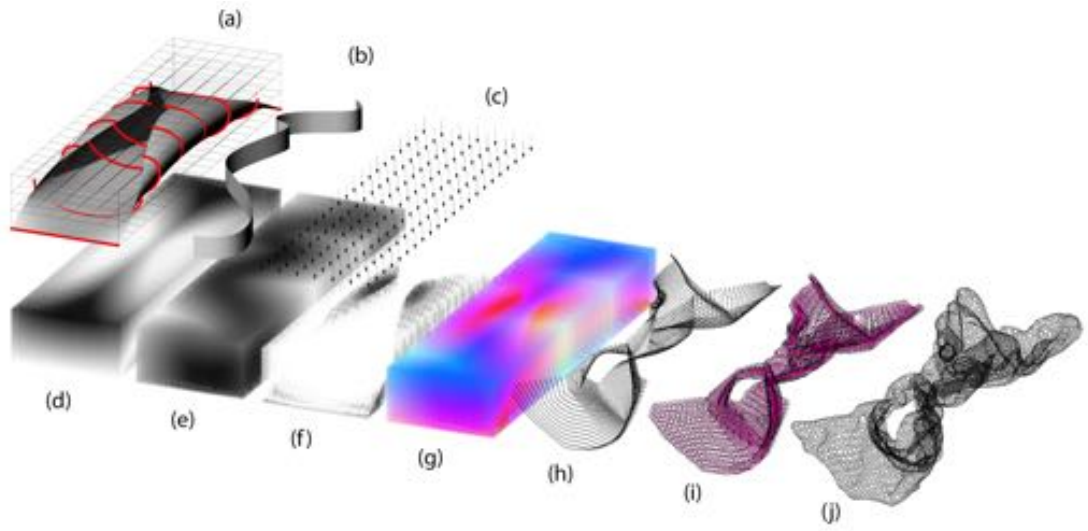


Figure 29: Data to weave workflow: (a) structural geometry; (b) architectural program; (c) simulated environmental data; (d) voxel grid: structure; (e) voxel grid: program; (f) voxel grid: light; (g) weighted voxel grid; (h) points on isosurface; (i) coarse mesh; (j) Kagome weave.

additions or cuts to the overall structure to adapt geometrically or programmatically to the local characteristics of the site (Figs. 28b and e).

It is also important to allow the designer to make top-down decisions during the digital design of the Kagome-based plant-scaffolds (Fig. 28c and f), to test and compare different scenarios and structures for their respective performance, visual and spatial characteristics in order to make informed decisions during the iterative design process.

Data-informed voxel grids are another approach to inform a design proposal for the woven plant-scaffolds. The voxel grids divide the design-space of the structure into finite building blocks (Fig. 29d to g showing the center points of the blocks). Each of the blocks or voxels can hold data-values (here: values used to color points from white to black) to inform the design in the step from graph/line to Kagome master surface, which we describefurther below. (Fig. 32, a to b). The grids can have different resolutions and hold either environmental data measured on site (Fig. 30a) or simulated data (Fig. 29d to f, Fig. 29b). The data can hold information about light (Fig. 29f, Fig. 30b), temperature, wind, noise, surface colors or materials (Fig. 30a), programmatic needs (Fig. 29e), structural needs (Fig. 29d) etc.

The data can be used to charge the voxel grid or the lines from the graph positively or negatively to generate isosurfaces which build the basis for the pattern generation of the weave (Fig. 29h). The charge values can be normalized and function as a threshold value for the creation of the surfaces. Different weights for different data sets can be applied to give the designer the option to change the importance and the influence of the data on the overall structure (Fig. 29g).

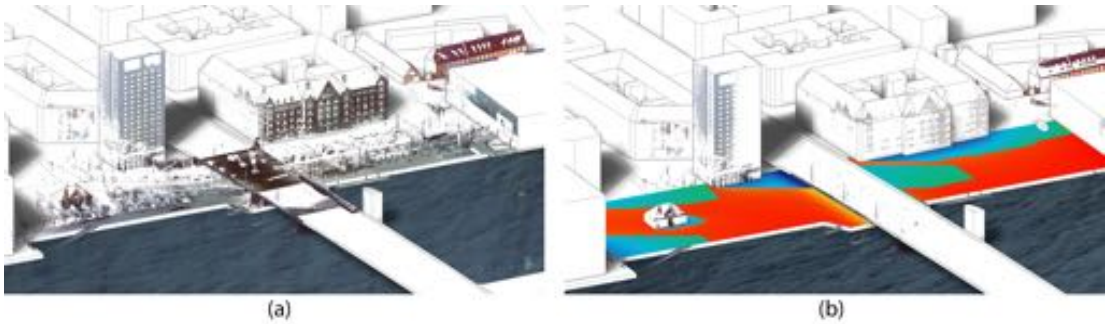


Figure 30: Site information and simulation: (a) surface materials and colors from 3D scan on site. (b) light simulation.

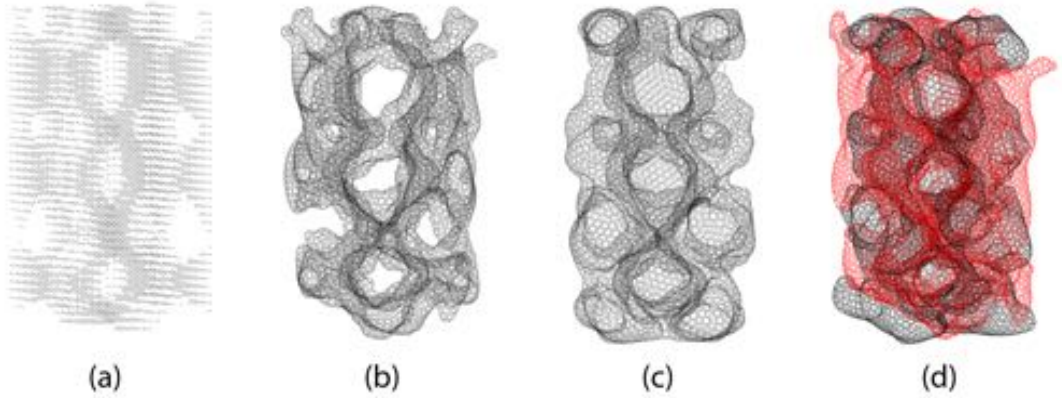


Figure 31: Charged data points to Kagome weave: (a) Data-informed charged points. (b) Weave from positive charge. (c) Weave from negative charge. (d) Weave from charge combination.

The weaves can be either generated from the positive or negative charged value points or a combination of both (Fig. 31). The following elements can be used alone or in combination as the input for a marching cube algorithm [14] to create isosurfaces in the design space:

- Weighted lines from the graphs.
- Drawn lines from the designer.
- Center points of the voxel grids with threshold value.
- Points drawn by the designer.

The resulting surface is triangulated (meshed, Fig. 32b). The resolution is changeable and has influence on the density of the final weave and its structural behavior / performance. The triangulation of the surface is based on a general mesh vertex valence of six with included valence singularities of five (or lower) for positive Gaussian curvature and seven (or higher) for negative Gaussian curvature [1]. An initial mesh relaxation equalizes mesh lines, angles and solves intersection problems (Fig. 32c).

The pre-relaxed mesh builds the basis for the identification of the individual weavers / filaments (Fig. 33, red and black curves) in the Kagome-weave. At the intersection points of both

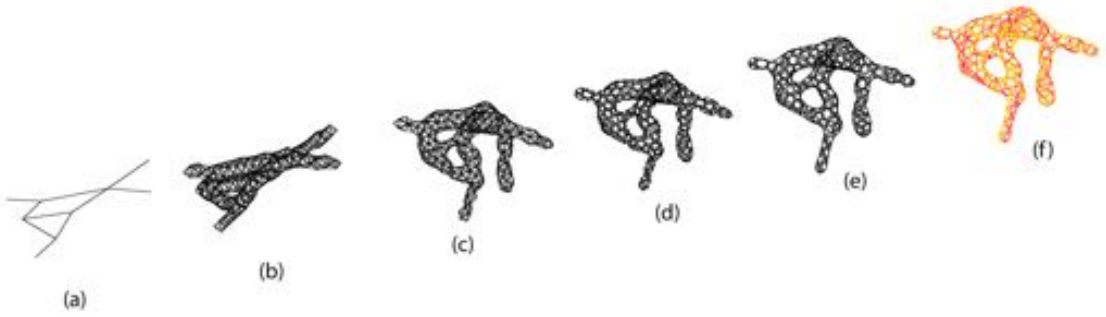


Figure 32: Line to Kagome weave pattern: (a) Lines as input; (b) isosurface / coarse mesh; (c) pre-relaxed coarse mesh; (d) up- and down-weaver identification; (e) relaxation Kagome-weave; (f) identification of filament orientation for post processing.

curve sets the curves subsections for one set are moved in the normal direction of the base mesh (Fig. 33, blue arrows) and moved in the negative normal direction of the other curve set. The resulting curves (Fig. 32d) are relaxed, where each weaver/filament tries to become a straight member, which creates the necessary interlacing of the system (Fig. 32e). The final Kagome weave is represented as a set of lines which can be used for further geometric analysis (Fig. 32f). This provides the basis for the iterative structural/plant growth simulation design feedback loop.

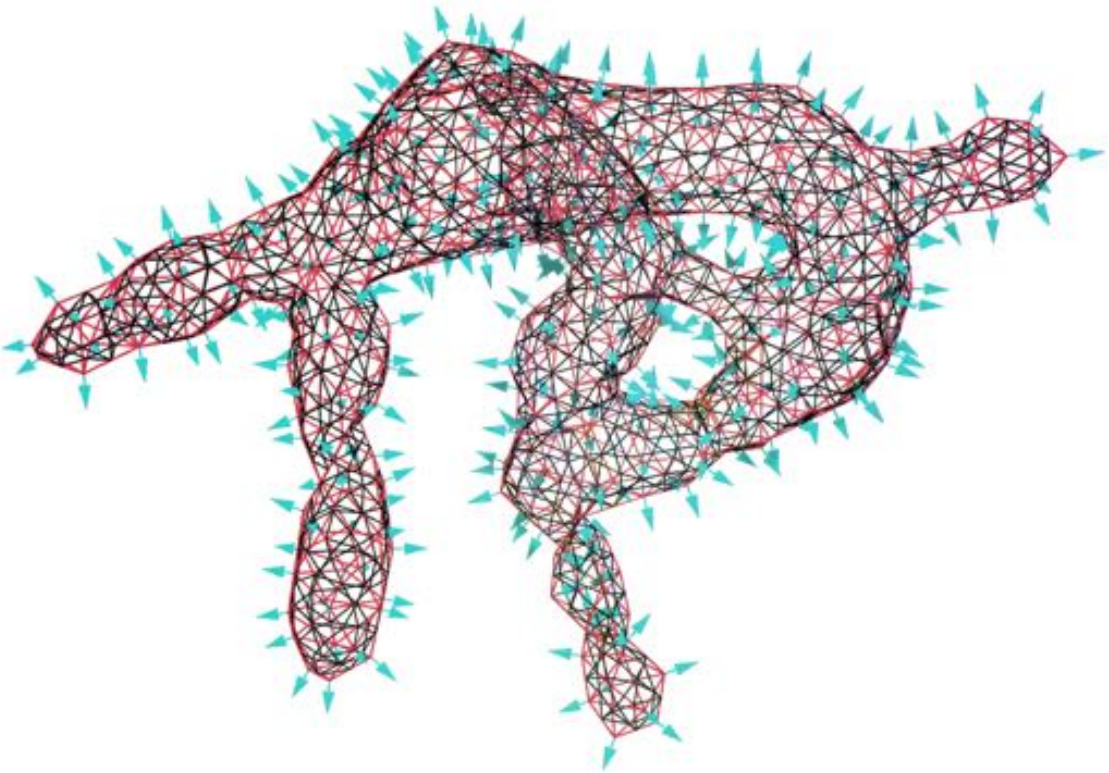


Figure 33: Alternating offset of up- and down-weavers in the mesh.

Indicative site & surroundings. The external urban plaza of the recently completed Blox building in Copenhagen has been chosen as a site for our hypothetical proposal. This site was selected because there are efforts by other parties to turn this area into an 'urban laboratory' for experimental installations that engage digital technologies for 'improving urban life.' We concentrate design efforts on the area located behind the building (Fig. 34a), but have identified further sites of interest along the harbour front (Fig. 34b to e) for later colonisation.

Figure 34: Sites of interest in the urban context around Blox.

For site A, we propose a canopy structure to provide a porous cover for the winter and natural shading through leaf cover in the summer months to protect from UV and help mitigate local urban heat island effect due to the hard artificial landscaping that has been introduced (Fig. 35).

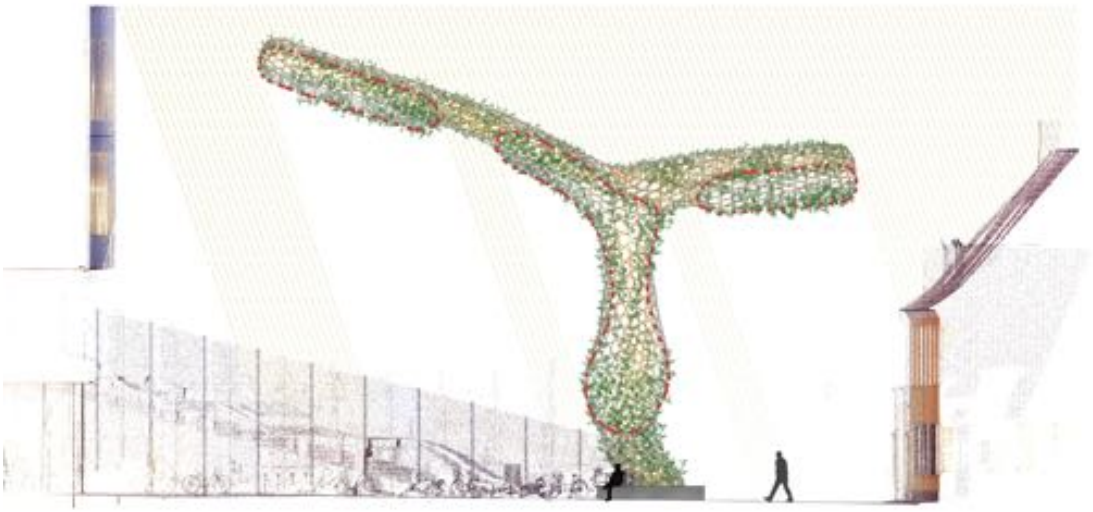
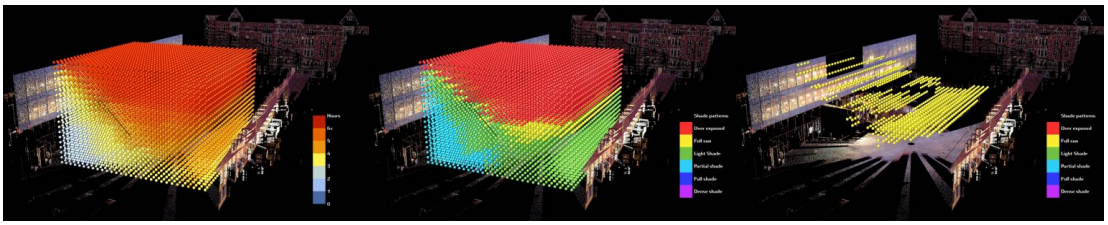


Figure 35: Section of proposed Living Weave canopy structure.

Combining intention & emergent self-organization. The character of the proposal develops out of the intention to provide a lightweight canopy that can be colonized by plants and produce habitats for other species. The geometry of the canopy is derived through a preliminary sunlight hour analysis to determine an ideal growth surface (Fig. 36c).



(a) (b) (c)

Figure 36: (a) Initial sunlight hours analysis; (b) sorted test points based on light requirements for different plant species; (c) isolated 'full sun' test points.

The Rhino/Grasshopper plugins, Ladybug and Honeybee [10], are used to analyse sunlight hours within a digital model of the target site. The analysis data is used to inform voxel grids about the existing local climatic context and to evaluate the interaction between design proposals and micro-climatic conditions in the early design phase (Fig. 36).

The sunlight hour analysis is set up as a voxel grid-based simulation (one-meter grid size) where the center point of each voxel becomes a test point for the simulation. To reduce computation time, the analysis period of the simulation has been limited to a single date (May 14 in this example). This allows for fast exploration and flexibility in the development of the workflow. After running the analysis, the results and corresponding test points are divided into separate lists based on light requirements for different plants:

- Dense shade: No direct sunlight.
- Full shade: Less than one hour.
- Partial shade: 1-3 hours.
- Light shade: 3-5 hours.
- Full sun: 6 hours.
- Over exposed: more than 6 hours.

The individual data sets can then be studied by the designer to make informed decisions about initial design intentions for the site. In this case, the test points corresponding to the data set 'full sun' was selected as an input to create an isosurface defining the early geometry of the intended canopy structure (Fig. 37b).

Once established, we generate connecting structure using the Vascular Morphogenesis Controller (VMC) growth algorithm [16]. The algorithm takes the results of the sunlight hour analysis as an 'environmental input to drive branching and connecting conditions (Fig. 37c). The data sets can be weighted differently to meet specific requirements for desired plant species as well as a driver to explore different spatial qualities of the resulting growth structure. The resulting graph of the VMC is then skinned with a triangluar mesh from which the Kagome weave pattern is then derived (Fig. 37e). The weave pattern generation provides fabrication information and is used as input into subsequent simulation and analysis steps to determine necessary and ideal plant growth paths, and to assess the 'growth career' of the bio-hybrid system.

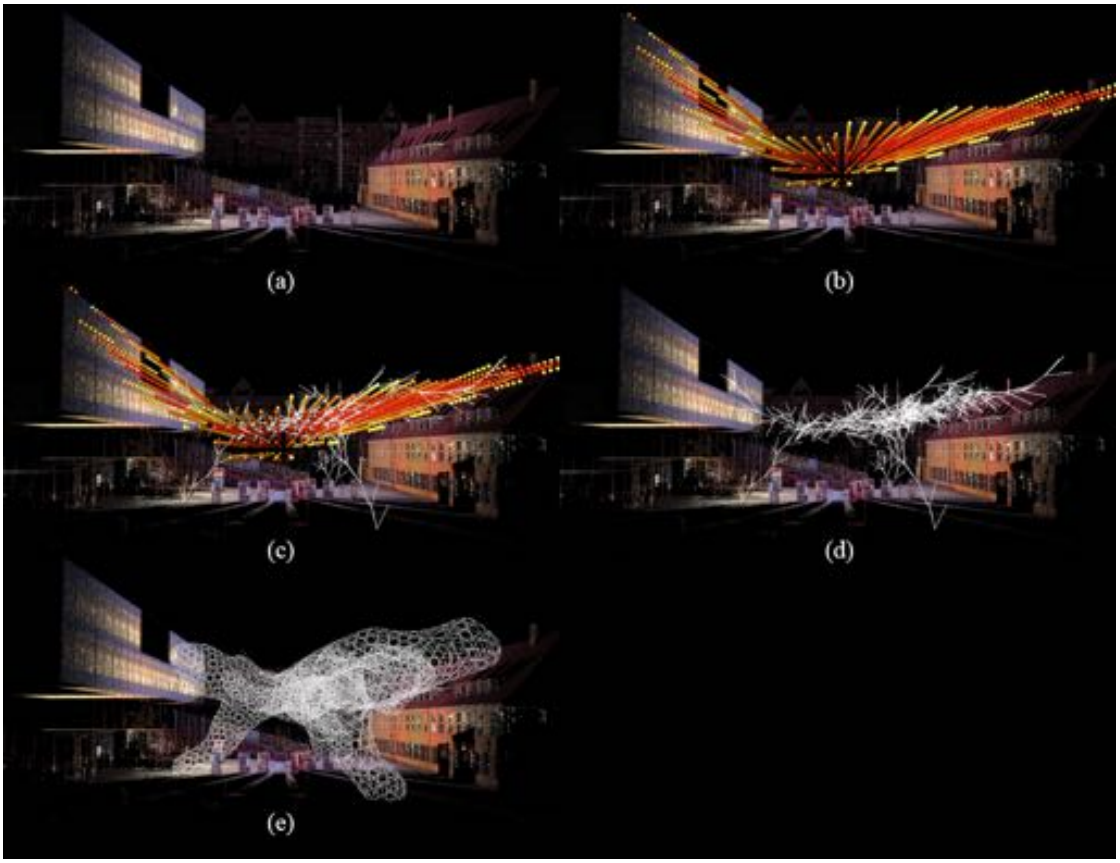


Figure 37: Iterative design workflow.

4.2.3. Performance informed iterative simulations

Plant growth is a source of continual change. As such, we need to establish tight iterative cycles of analysis and simulation in order to study emerging design opportunity, changes to mechanical and environmental characteristics and explore the consequences of design decisions across varying time periods. In the proposal, we focus attention on the simulation of plant growth across the designed structure and analyze changing structural performance and aggregated daylight/self-shading conditions to determine the specification of ideal growth paths and spatial conditions through plant cover and density.

Finite element modeling. In order to analyze the changing structural performance of hybrid structures, a parametric finite element model is developed using the Rhino/Grasshopper plugin Karamba3D [9]. The simulation process is described in Fig. 38 for a simple planar condition subject to its own self-weight, but incorporating plant growth as tri-axial members of the weave structure.

The geometry consists of wood strips and climbing plants interlaced in a sparse triaxial weave (Fig. 38a). The geometry of the indicative plants is generated using a simplified growth model in interaction with a dynamic simulated light environment. Plants and weaves are discretized as beam elements with circular and rectangular cross-sections respectively (Fig. 38b). Materials are assumed homogeneous and isotropic. The contact interactions between the weave members are defined as spring elements with a variable stiffness in translation (Fig. 38c). This stiffness at the interaction with the plants is also expected to evolve during the growth of the plant, though it is neglected here. For simplification purpose the dead weight of the overall structure is the only load considered (Fig. 38d). No initial stress resulting from the interweaving process is considered. A static linear analysis performed on the hybrid panel provides the distribution of the axial stress and displacement in the plants and the weaves (Fig. 38e and f).

The mechanical properties of the materials are obtained from three-point bending tests carried out on different samples of the Maple strips used in the bi-axial scaffold, and samples of Fallopia at different stages of growth (Fig. 39 and 40). While they are supported by the strips at earlier stages of growth, the plants progressively reinforce and contribute structural capacity back to the weave as shown through the analysis in Figs. 41 and 43. The same process can be applied to more complex woven geometries as shown in Figs. 42 and 43. The use of such simulations also helps make a coarse approximation to determine the minimum growth requirements of the plants in specific areas of interest. As the following proposal deals with a more complex and larger scale structure, it was chosen to simplify the modeling computation time. The plants are assigned the same geometry as the other weaves and equivalent mechanical properties are defined to account for their changing performance with time.

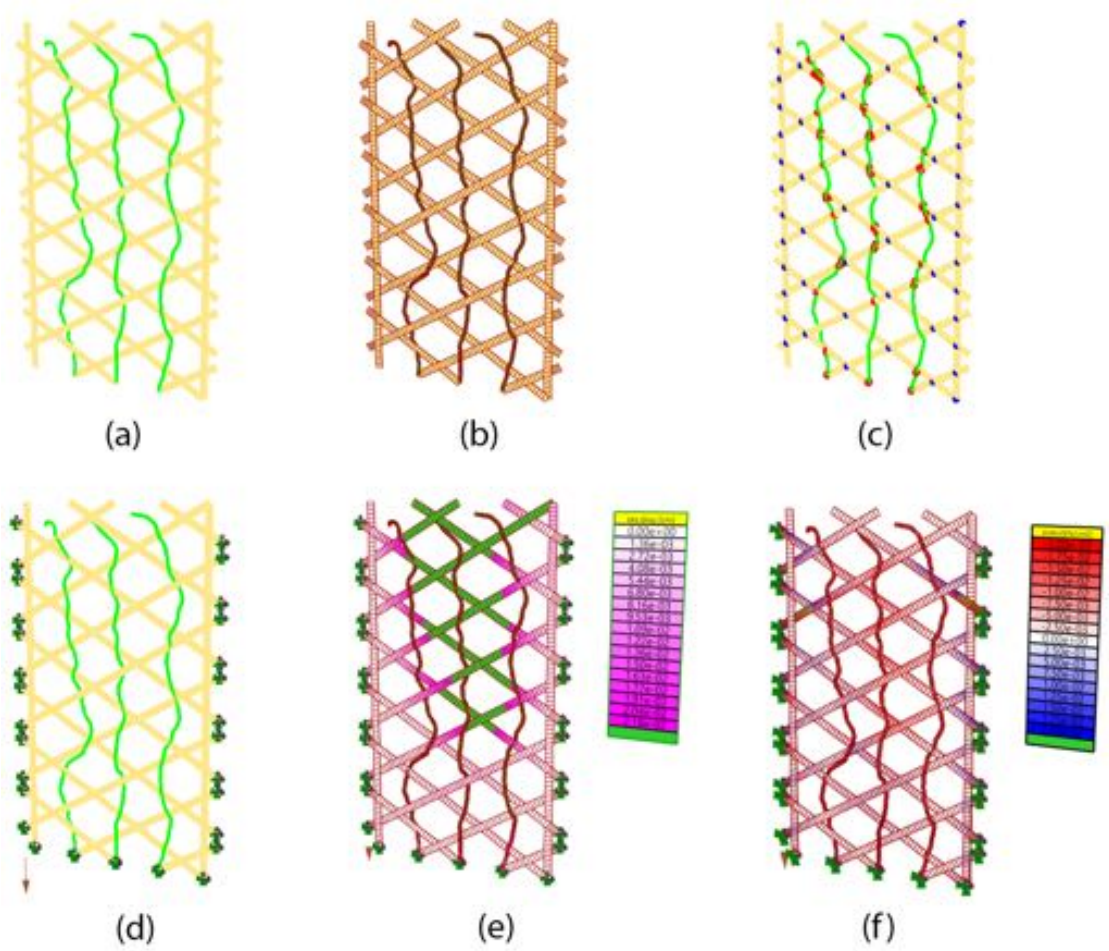


Figure 38: Simulation process on a hybrid weave panel: (a) geometry, (b) discretization in beam elements, (c) contact interaction, (d) loading and boundary conditions, (e) displacement, (f) axial stress.

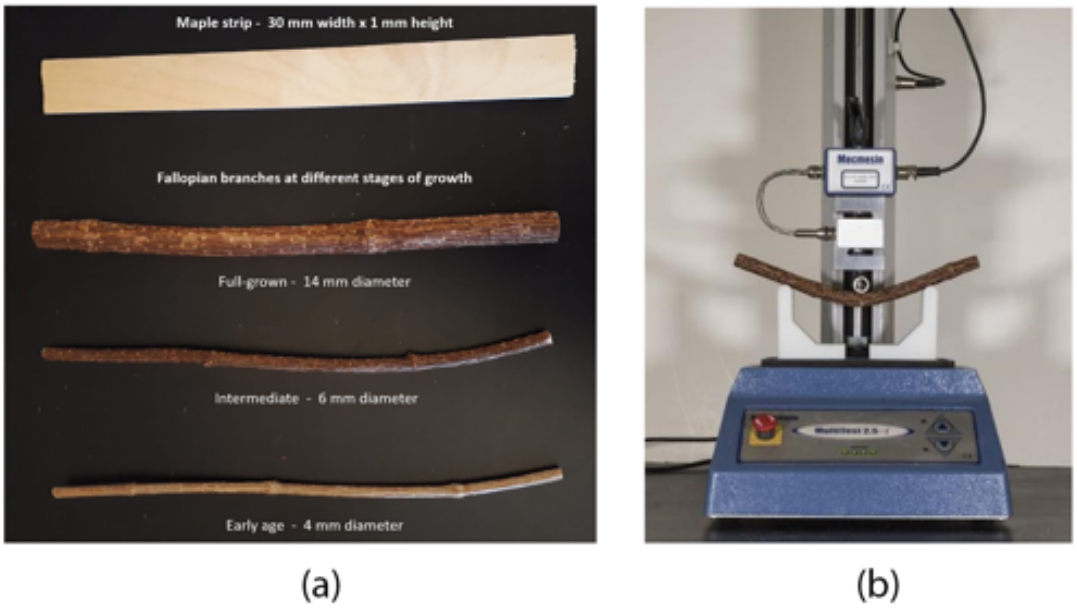


Figure 39: (a) Samples of Fallopia vine at the different stages of growth and Maple used for mechanical characterization; (b) three-point bending experimental setup.

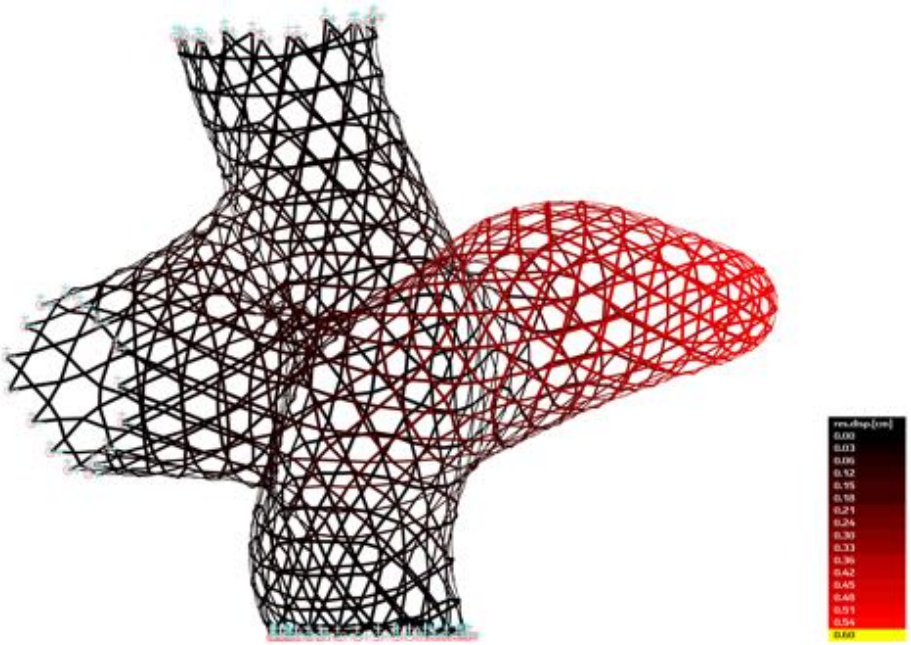


Figure 42: Displacement distribution on more complex woven geometry.

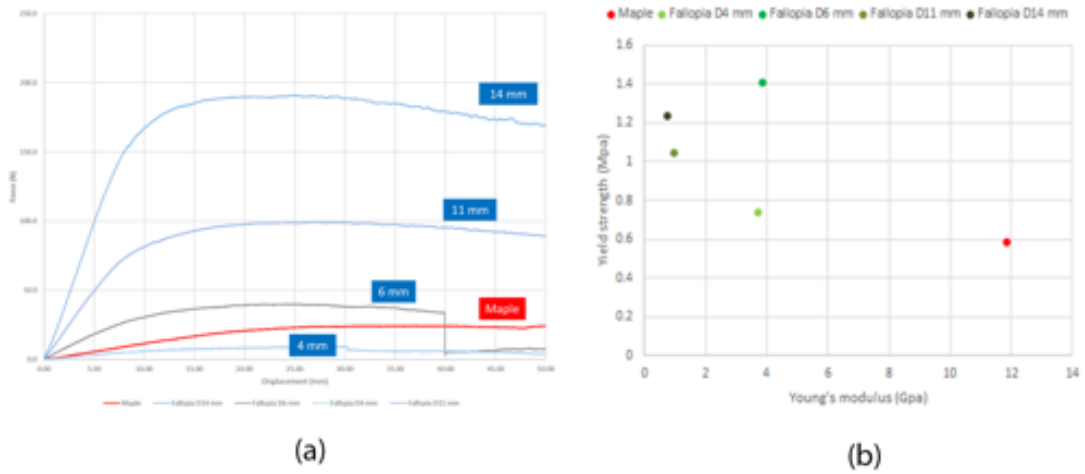


Figure 40: (a) force-displacement curves for Fallopia vine at different stages of growth and Maple samples tested in three-point bending experiments; (b) flexural strength vs Young's modulus for the Maple and Fallopia branches.

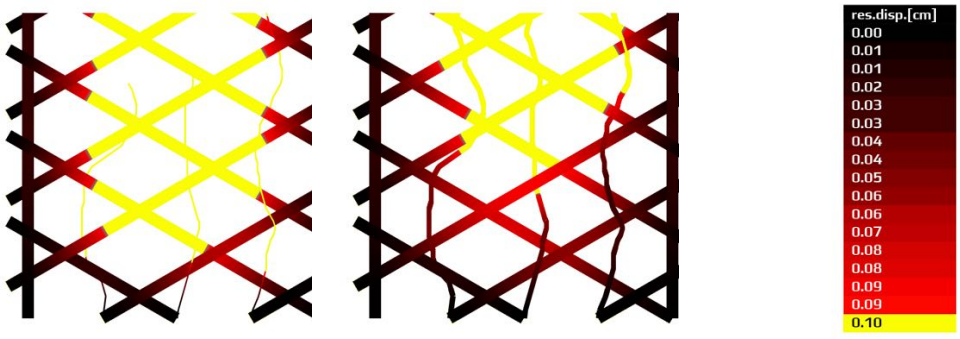
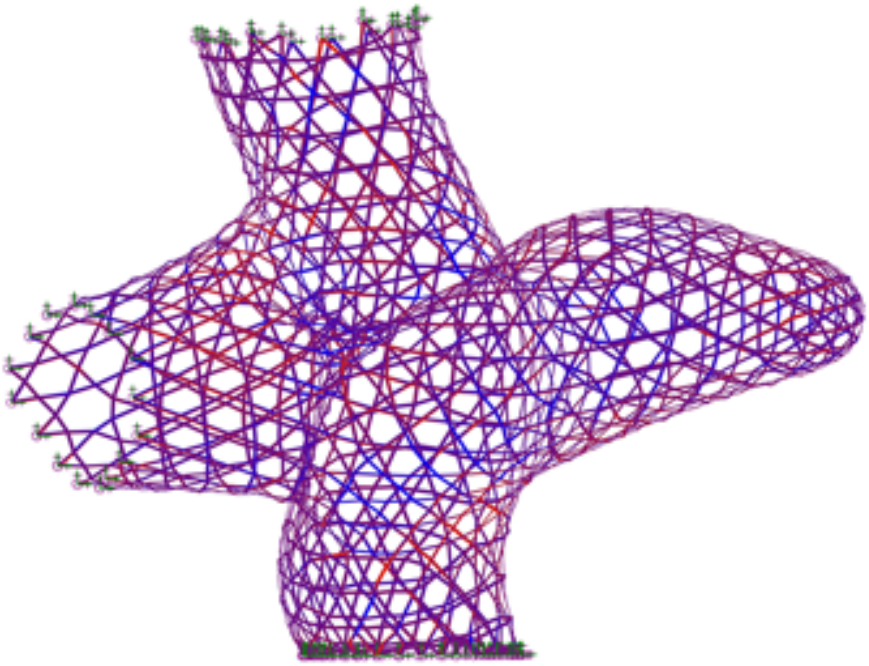
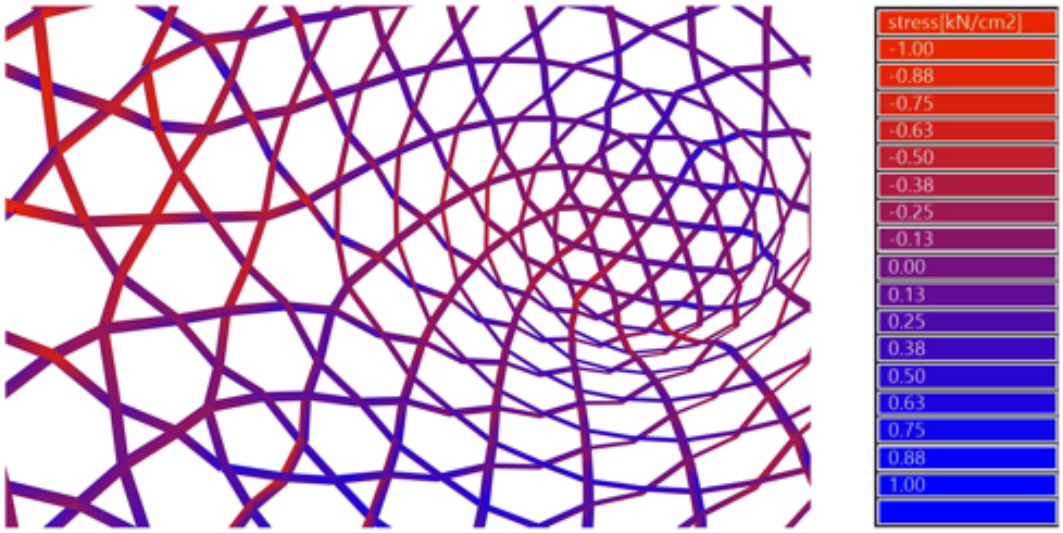


Figure 41: (a) Displacement of weave in early plant growth stage and; (b) later plant growth stage.

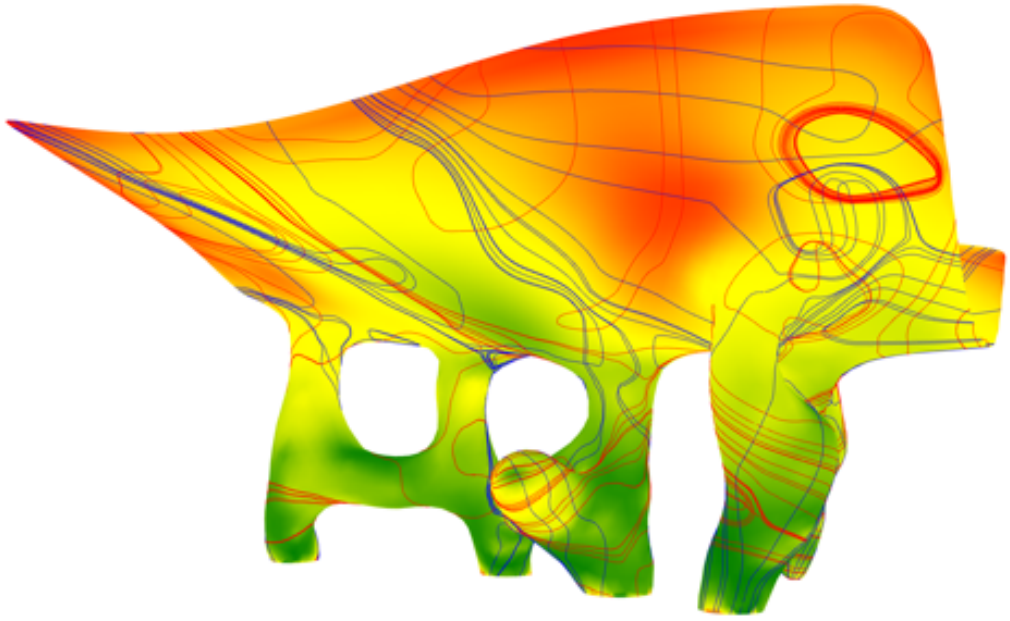


(a)

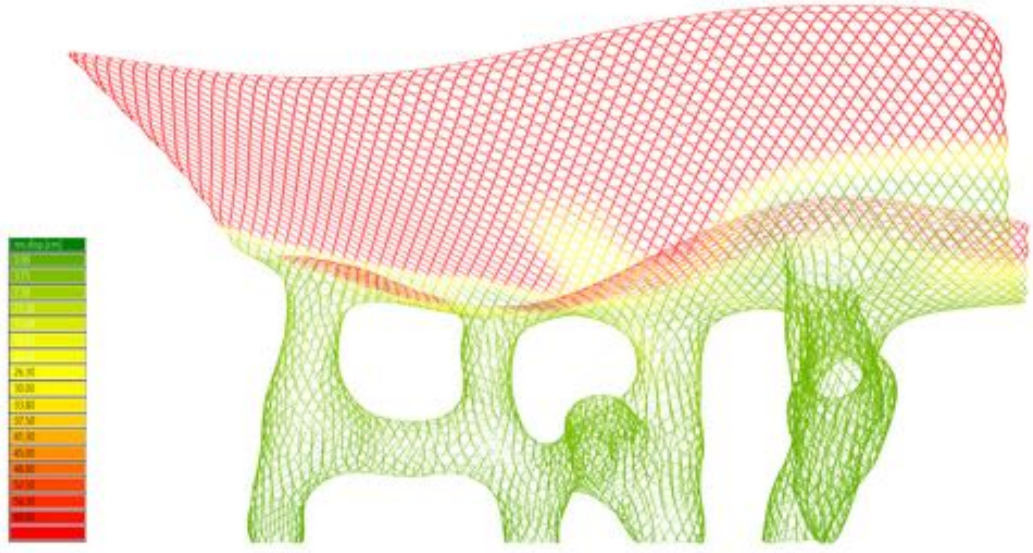


(b)

Figure 43: Compressive (red) and tensile (blue) axial stresses on more complex woven geometry: (a) overall structure, (b) detailed view of weaves from inside the structure.



(a)



(b)

Figure 44: (a) Principal stress lines represented on a preliminary shape of the canopy; (b) displacement distribution in the weave geometry at a later stage of design.

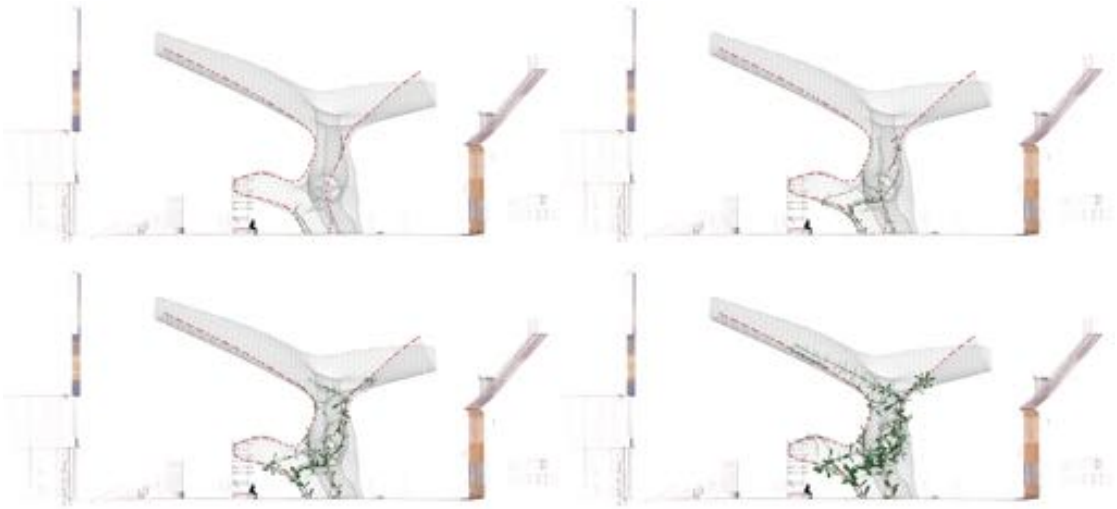


Figure 45: Plant growth pathways informed by structural analysis.

4.2.4. Pathways for plant growth

In order to identify ideal pathways for plant growth, structural simulations are performed on different iterations of the considered structure. Plants are intended to replace weave elements in the areas of interest where they will provide a higher structural capacity once they reach maturity.

Different strategies can be considered for the identification of these paths, depending on the maturity of the design at the stage these choices are made:

- It can be decided to grow plants along the stress lines of a preliminary shape of the structure. These lines represent the directions in which the efforts are transmitted. Once the weave geometry is defined these pathways are mapped to the nearest weave elements so they can be replaced by plants (Fig. 44a).
- Another option - which can also be complementary to the first one - is to strengthen the structure around the locations showing an excessive displacement or stress on the weave geometry (Fig. 44b).

Whatever the strategy adopted, several iterations are necessary to refine the choices made and ensure the structural integrity of the design at different stages of growth (Fig. 45). This is particularly important at earlier stages of plant growth to determine the position of temporary supports while waiting for plant material to mature to the required performance. Such supports can offer temporary architectural opportunity during this phase of the growth career, before being relocated or removed.

4.2.5. Analyzing conditions for species diversity

After identifying ideal pathways for plant growth, the hybrid weave is fed back into the sunlight analysis tool to determine the number of sun hours across its surface so that specific conditions can be mapped to ideal supplementary species that can provide other architectural qualities. In combination with structural analyses, this allows a developed understanding of suitable species

to achieve specific performance requirements, exploit intrinsic conditions created by the weave and support contextualised place-making (Fig. 46a).

Effects on surroundings. In addition to the analysis of sun hours on the weave itself, its impact on shading to the surrounding area is also considered (Fig. 47). This feedback can be used to refine geometry and geometric features, such as openings, to ensure ideal conditions for surrounding plantings and anticipated uses (Fig. 48).

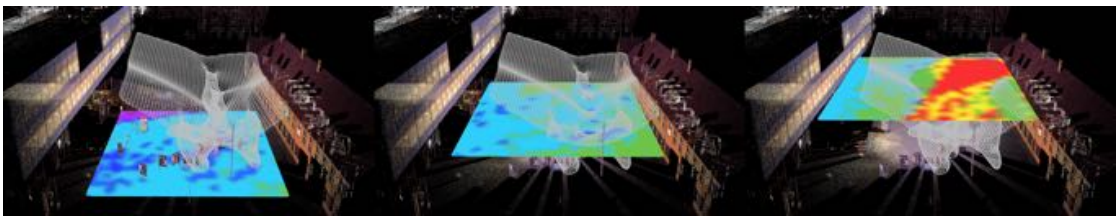
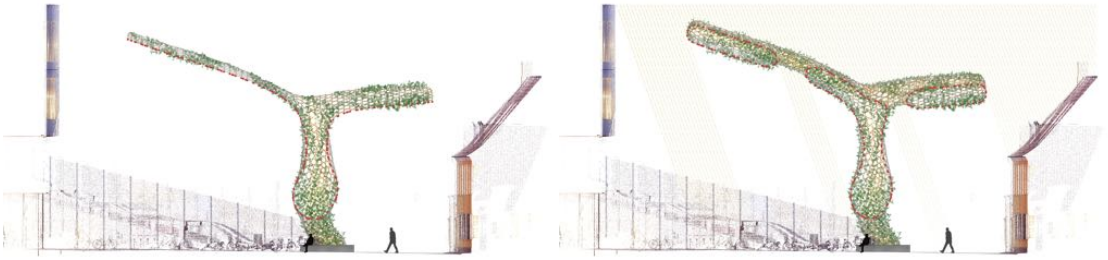


Figure 47: Evaluation of the interaction between design proposal and sun hours on site.



(a) (b)

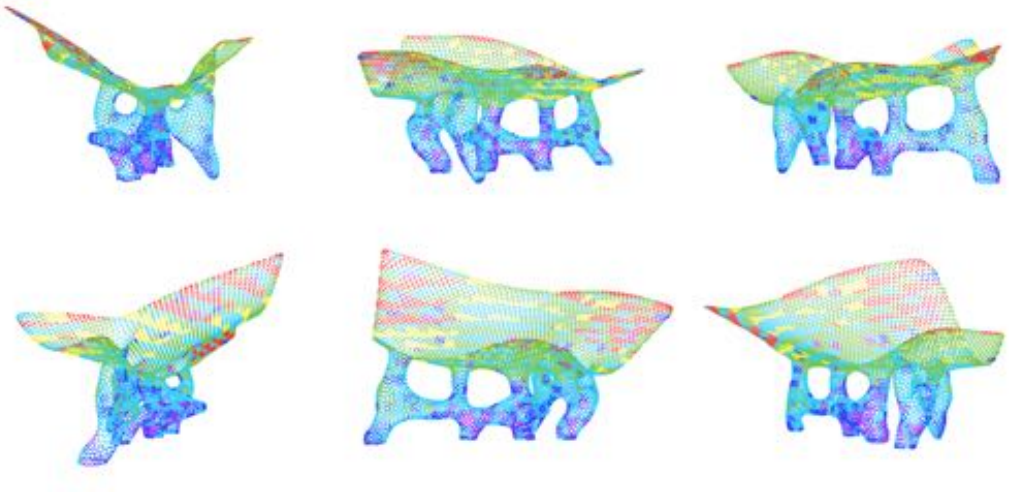
Figure 48: (a) Section of proposal in early stage design; (b) changes made to canopy structure after feedback information from structural and sunlight analyses.

4.2.6. Connecting plant communities

Here we speculate on the possibility of creating an artificial analogue of the 'wood-wide-web' - the naturally occurring mycelium based network that binds individual plants into a communicating community. The feasibility of this is based on Cybertronica's MU system and its interactions reported above in Sec. 2.1.3.

Deploying this technology in urban contexts would allow the networking of disconnected plant communities (Fig. 49) to enable environmental monitoring, regulation of environmental controllers (such as supplementary urban lighting and water misting) and promote long-term adaptation and learning.

This will create engaging, resilient, adaptive, co-occupied and aesthetically potent public spaces aimed at increasing bio-diversity and city live-ability for complex living ecosystems. Fig. 50 indicates the connection of existing trees isolated by hard landscape, now connected and communicating with the bio-hybrid system.



(a)

(b)

Figure 46: (a) Sun light hours on weave; (b) plant species categorized by light requirements.

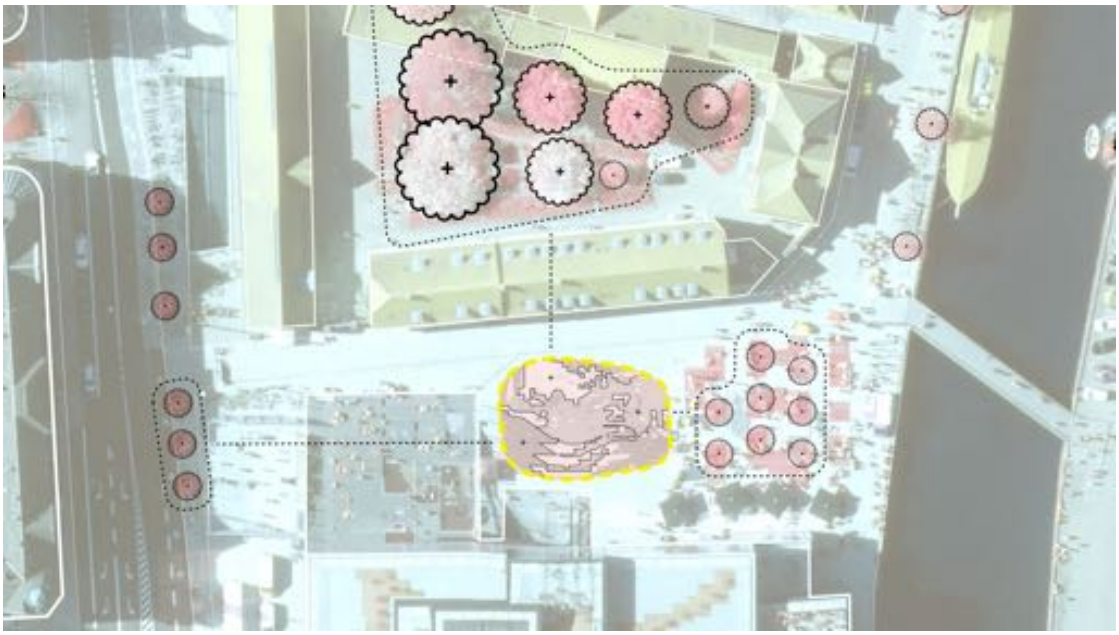


Figure 49: Networking of disconnected plant communities.

4.2.7. Conclusion

In this section we have presented an urban proposition for a *flora robotica Social Garden*. The proposition draws upon the scientific insights and technologies developed throughout the research project to inform a speculative design concept. The concept demonstrates the symbiotic integration of plant and artificial material to grow a living architecture that couples natural and artificial growth, and demonstrates an approach to combining predetermined design intent with processes of natural and artificial self-organisation. We have also reported on the design workflow that has been developed to support design development and evaluation. The proposition limits itself to considering a fixed scaffold that adapts over time through directed growth of natural plants. Exploring the architectural implications of working with morphologically adaptive artificial scaffolds remains an open opportunity, but one that has, in principle, been proven to be technologically viable (see the following section 5.2 *Artificial Growth*). In addition, the experimental work by partner Cybertronica on the usage of plants to perform functions, promises to have significant architectural implications, but remains an under-explored design territory within the work presented here.

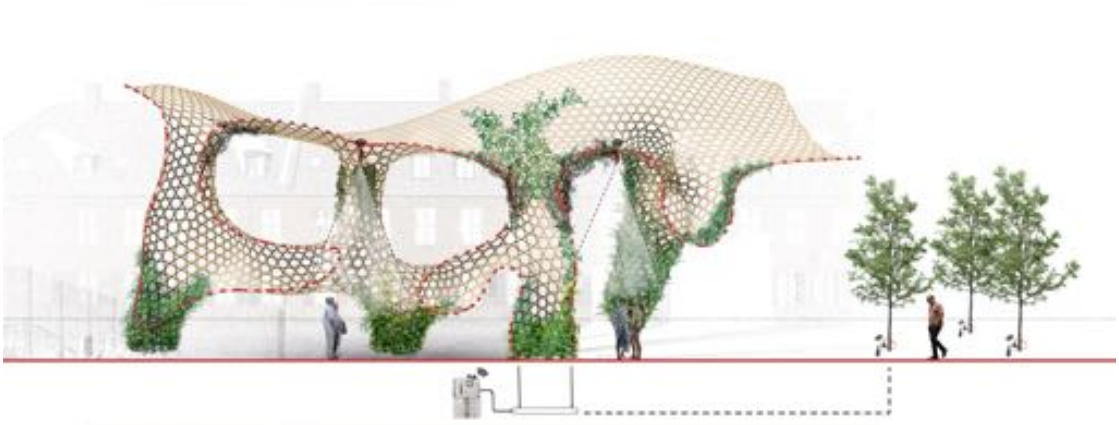


Figure 50: Bio-hybrid network.

5 Dynamics of growth for social gardens

5.1 Mechanics and overall experiment setup for bio-hybrid growth

This section describes the mechanics and overall lab setup developed to fulfill the experiment design laid out in Sec. 4.1.

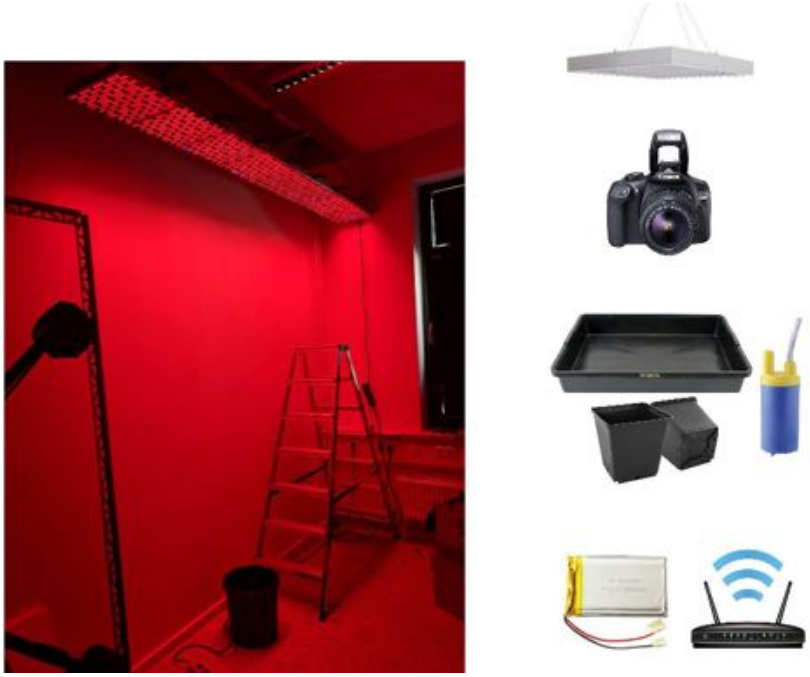


Figure 51: (left) The overall lab setup. Commercial growth lamps produced for greenhouse conditions, with only Red LEDs visible, hung at ceiling. (right, from top to bottom) Supplemental Growth lamps for hanging at lower heights; DSLR camera for timelapse photography; system for fully automated watering; Local Area Network and back-up battery setup allows system to continue running during a disruption such as power outage.

In this experiment setup, a full indoor lab room is converted into a room with the desired conditions for plant growth. The Global Environment Monitoring (GEM) system developed and reported in previous deliverables is again used here, with updates for increased reliability. Further improvements to the lab setup (see Fig. 51) compared to previous setups are as follows. The level of red light is dramatically increased, for support of plant photosynthesis. Red LED growth lamps are hung from the ceiling along the full length of the experiment setup, and are hung on stands at either side of each module, along the full height of the growth zone. The quality of experiment documentation is improved, by using DSLR cameras for timelapse photography, rather than lower-resolution Raspberry Pi cameras. The watering system is improved by transitioning to full automation with soil-moisture sensors and pumps from a central water container, rather than partial automation via sensing and manual watering. The powering of the network setup is also improved. If wireless connection is temporarily disrupted, the system can continue to proceed with the experiment and documentation via the Local Area Network, and only uploading of recordings will be delayed, proceeding when connection resumes. Likewise, if there is a power

outage or similar, the essentials of the system can run on back-up battery power until power is restored, circumventing corrupted SD cards or other substantive failures.

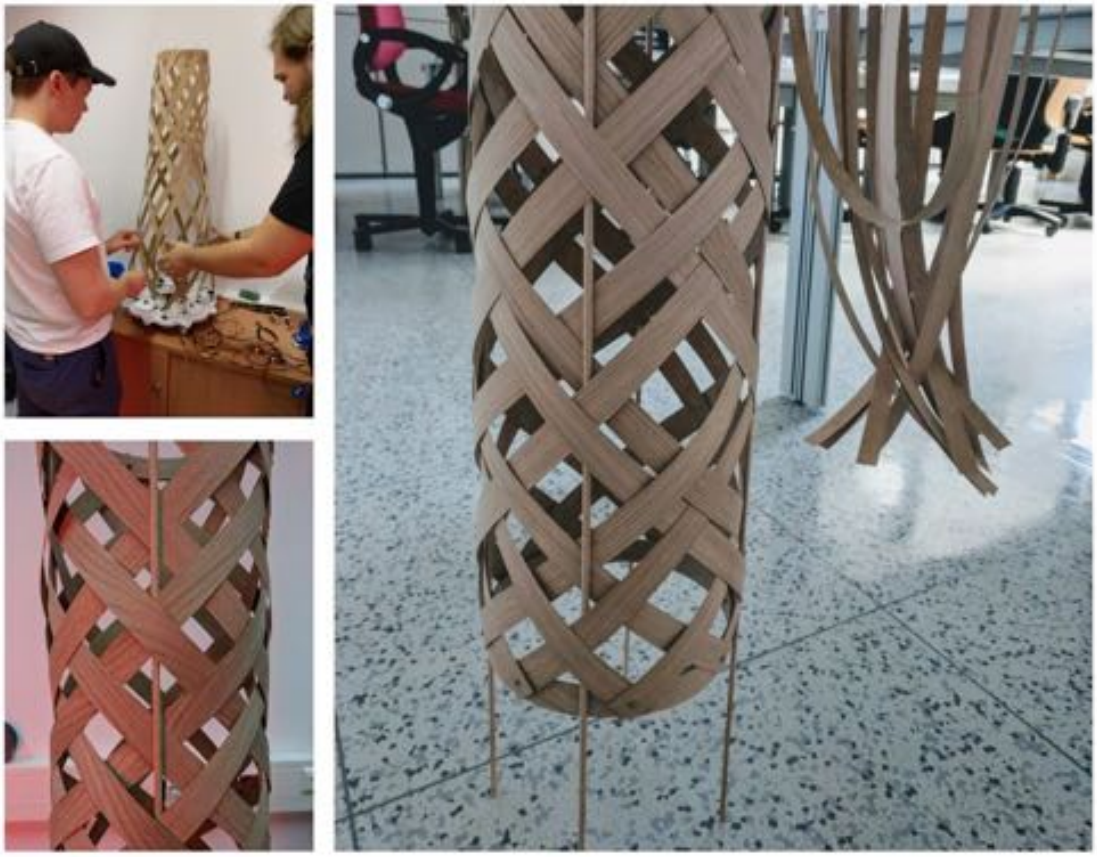


Figure 52: One braid tube for a 4-braid scaffold module; (bottom left) a 1-by-1 pattern wooden braid, reinforced with triaxial wooden rods; (right) a 2-by-2 pattern wooden braid, reinforced triaxial by wooden rods, standing next to loose wooden strips that the braid is made from. This braid is self-supporting—it is standing on the wooden rods resting on the ground. This braid was manufactured by the braid machine; (top left) the braid machine in process manufacturing that braid.

The material used to execute the planned scaffold is a commercial wood product manufactured for edgebanding. The banding is reinforced with a fiber backing, meaning that if a crack forms in the wood of the strip, it will not fail, as the fiber reinforcement will act in tension to hold it together. Here we use hardwood banding, specifically oak, to achieve slightly more stiffness and less brittleness than a softwood banding. The banding is the ideal material, because it is wide enough and stiff enough to construct self-supporting braids when reinforced with a few wooden triaxial rods, but is flexible enough to be braided by the braid machine (see Fig. 52). In addition to allowing a self-stable hollow tube braid, the use of strip material (as opposed to rods) allows for flexibility in attachment approach for the robotic nodes (see Fig. 53). The material also enables relatively flat and seamless integration of the second braid layer combining tubes (see Figs. 54 and 55).

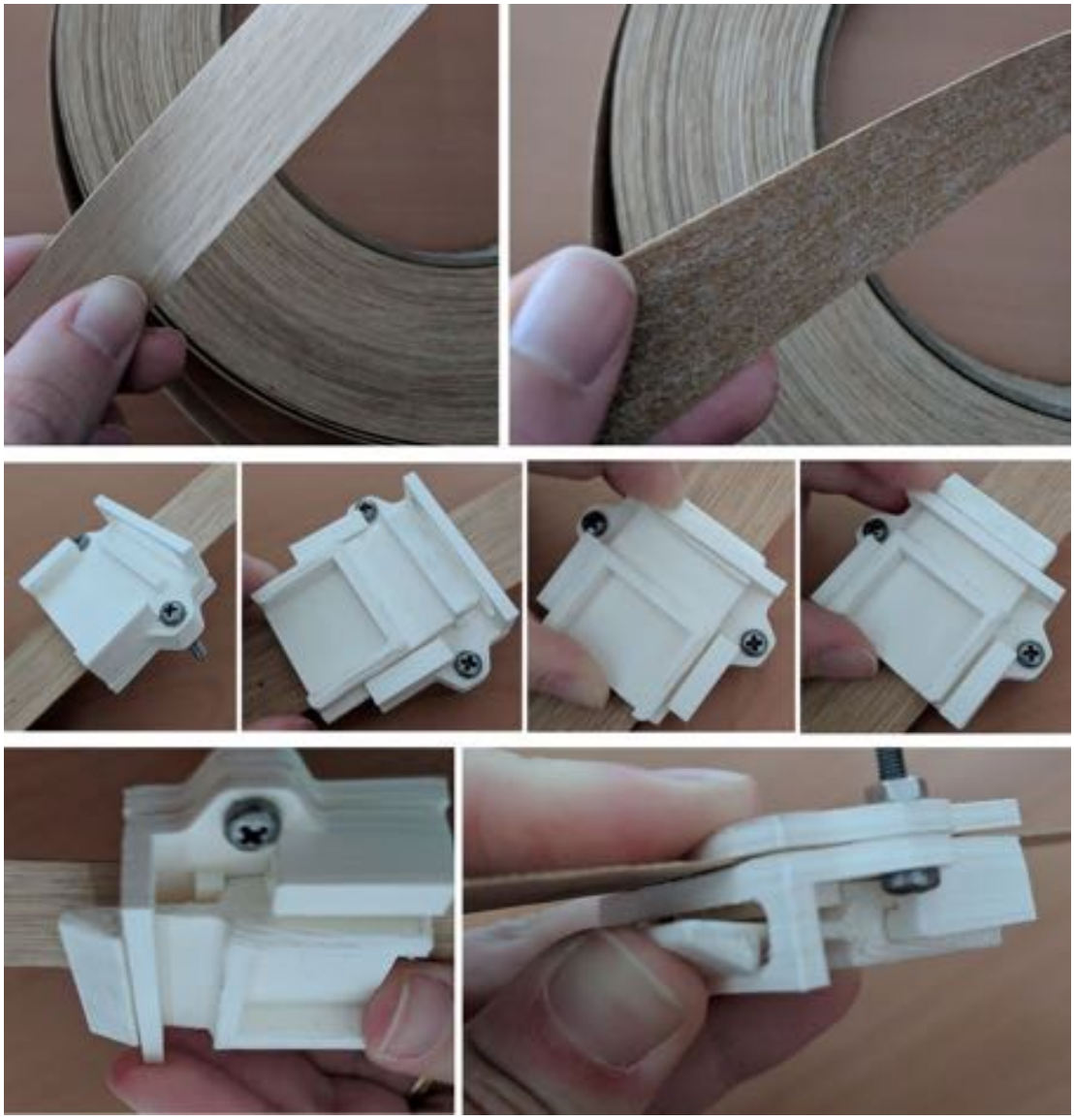


Figure 53: (top) The material used is wooden strip reinforced by fiber backing; (middle, bottom) Isolated 3d-printed parts showing snap-fit connection between the wooden strips and another element, such as the robotic node for plant shaping. The connector show here has three parts. Two of the parts are fitted around a strip, and are screwed to one another to fasten to the strip via pressure and friction. This avoids making holes in the strip, which may limit the reconfigurability of the system or in extreme cases may cause structural damage to the braid.

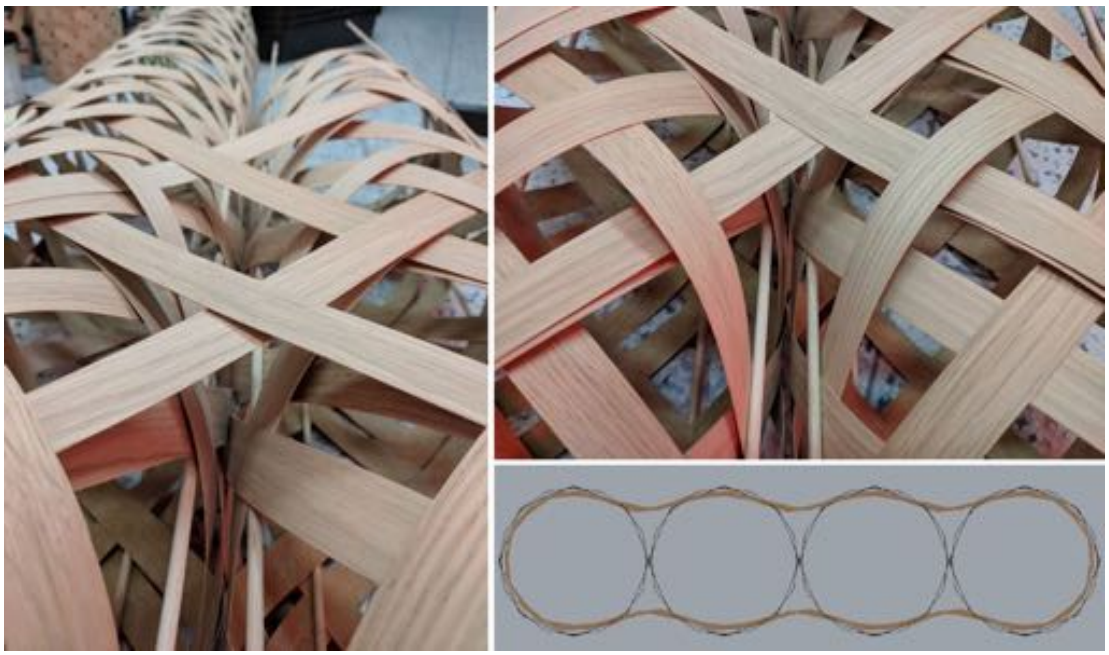


Figure 54: The individual braid tubes are combined by a second layer of braid that bridges the gaps between them.

Figure 55: An example double-layer 4-braid module, fitted in the lab setup with an unenclosed robotic node for plant shaping.

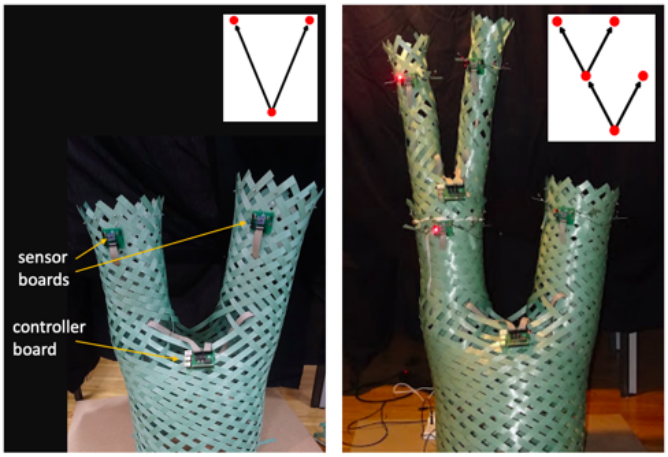


Figure 56: An example braided module (left), and two connected modules (right), with their overlaid VMC graphs (inset images).

5.2 Artificial growth

Braided structures growth guided by VMC on raspberry Pi network. A set of experiments for growing braided structures guided by their embedded VMC are presented in the following.

Physical setup. The building blocks of the structure are Y-shaped braided modules each augmented with a controller node and a set of sensors. A controller board is mounted on the braid and is connected to two sensor boards each mounted at a branch of the module and containing four light sensors and an accelerometer. The modules can be connected to each other by attaching the bottom of a module to a branch of another module making the former module a child of the latter (see Fig. 56). The communication between the controller boards of the connected modules is maintained via the sensor boards of the parent module. Once a branch is ready to grow or to be removed, it signals the humans via the LEDs located at the sensor boards. The human in turn manually realize the change to the structure (addition/removal). A main VMC node is running on each controller board. If a branch of a module is not connected to another module, the controller additionally holds a leaf VMC node associated with the free branch as a child of the main node. Otherwise, the main node of the module connected to the branch is adopted as the child node of the parent module. The VMC graph is therefore the network of the connected VMC nodes distributed over the physical structure. The nodes have access to local sensors and transfer the flows to their children and parents that may locate at other physical modules. Since the role of the leaves and the interior nodes are different, they might take different sensory information into account. For example, the leaf nodes may sense the local intensity of light and temperature in producing the successin while the interior nodes may sense the tilting of the module or the mechanical stress on the module to adapt the rate of successin transfer accordingly (e.g., to add a preference for growing branches under less mechanical stress due to environmental and structural conditions).

Parameter setup. In the following set of experiments, two input sensory variables are implemented, I_{light} and I_{tilt}. The input variable I_{light} measures the local light and is used at the

leaves and I_{tilt} reads the local accelerometer (indicating the tilt of the branch) and is used at the interior nodes. The parameters are chosen for the experiments based on the understanding of the parameter effects obtained from D2.3 and preliminary experiments adjusting the values for the particular conditions (i.e., lighting) of the setup and the planned experiments. The parameters may vary according to the focus of each particular experiment. The value of input variable I_{light} is the average of all four light sensors scaled to $[0, 1]$. The successin is produced at the leaves only based on the value of I_{light}. Therefore, $\omega_{\text{light}} = 1$ with no constant production rate (i.e., $\omega_{\text{c}} = 0$). Due to technical reasons [13] with respect to the implemented communication protocol, the values of the successin at all the leaves are rescaled with a factor of 0.167. The accelerometer-readings are also scaled to [0,1] to obtain the input variable I_{tilt} which influences the transfer rate at interior nodes. To compute the transfer rate, the input value is weighted by a ρ_{tilt} and added to a $\rho_{\text{c}} = 0.5$. In most cases in the following experiment, the measured value of $I_{\text{tilt}} \simeq 0.99$ and the $\rho_{\text{tilt}} = 0.5$, thus the transfer rate is $\simeq 0.99$. The different cases are explained within the particular experiments. In most experiment $\alpha = 0.9$ and $\beta_{\text{c}} = 2$ are chosen to allow fast adaptation and a medium competition respectively. In all experiments, $R_{\text{root}} = 1$.

Growing structures with different competition rates. Two different competition rates, $\beta_c \in \{1, 2\}$, are used in this experiment to grow two structures with identical environmental conditions. A light source is located at the top-left of the structures. The experiment demonstrates the different behaviors of the structure in terms of growing towards light. The threshold values for addition and deletion of the nodes are chosen $th_{\text{add}} = 0.25$ and $th_{\text{del}} = 0.2$ respectively so that only a limited number of options are available. When several branches are ready to grow, that is, when their resource is higher than the threshold, the user is free to choose the option they personally prefer. Fig. 57 shows the growth of the structure with $\beta_c = 2$. At each step of the growth, a new module is added to one of the leaf branches with resource higher than th_{add}. In case of several options, the user chooses their desired option – here the branch with the highest resource. Fig. 58 shows the growth of the structure with $\beta_c = 1$. Because β_c cannot have any influence on the behavior of the first single module, we started the experiment with a second module already connected (step A in Fig. 57). The figures show a stronger growth towards the brighter regions of the environment with the larger competition rate and a relatively small preference for light and more tendency for growing a symmetrical (bushy) structure with the smaller competition rate. Note that the difference in the behaviors for the two values of the competition rates is comparable in identical light conditions. That means, for example with a small competition rate, the structure could be more asymmetric than its corresponding structure here, if the light intensity in the environment was different.

Combined effect of transfer rate and competition rate. In this experiment the combined effect of transfer rate and competition rate is investigated. The final structure from Fig. 57 is used. Here, instead of the directional light source as in Fig. 57 (located on top left of the structure), the experiments are performed under ambient room light. The light received by the different leaves differ from each other due to reflections and shadows in the environment, but the variance is much lower compared to Fig. 57. The constant transfer and competition parameters are $\rho_{\text{c}} \in \{0.25, 0.5\}$ and $\beta_{\text{c}} \in \{1, 2\}$. Considering that $\rho_{\text{tilt}} = 0.5$ and $I_{\text{tilt}} \simeq 0.99$, then $\rho \in \{0.74, 0.99\}$.

[13] The communication protocol used here limits the values in the interval $[0, 1]$. Since successin from different leaves is added together on its way to the root, we need to define a scaling factor according to a maximum value allowed for the number of leaves in a structure. In the following experiments we set the maximum value to 6 (maximum 6 leaves in the structure) and therefore the scaling factor is set to $1/6 \approx 0.167$.

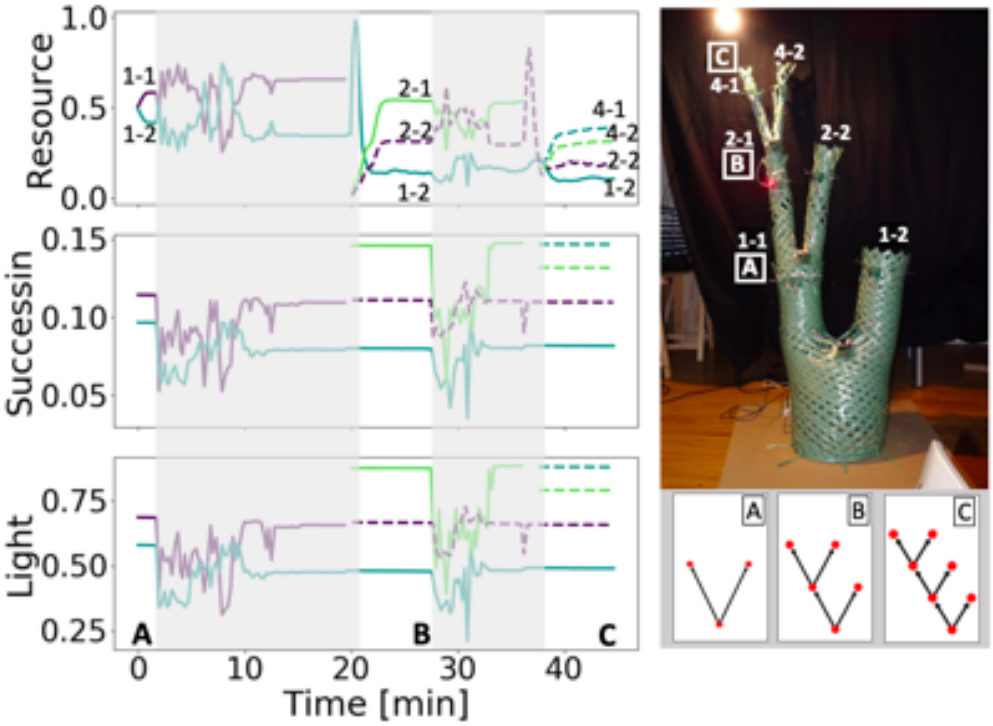


Figure 57: Growth with $\beta = 2$.The plots on the left demonstrate the values of resource, successin, and the light input for selected nodes over the course of the growth. The final structure is depicted on the right. The A-C labels in the plots mark the steps right before the start of manual growth. In the photo of the final structure, the labels indicate the position of growth at each step. The shaded parts of the plots indicate the periods when growth was physically realized. The labels 1-1, 1-2, etc. mark the location of the selected nodes on the structure and their corresponding values (resource, successin, light) on the plots. The graphs on the bottom-right depict the virtual VMC graphs running on the structure at each step of growth.

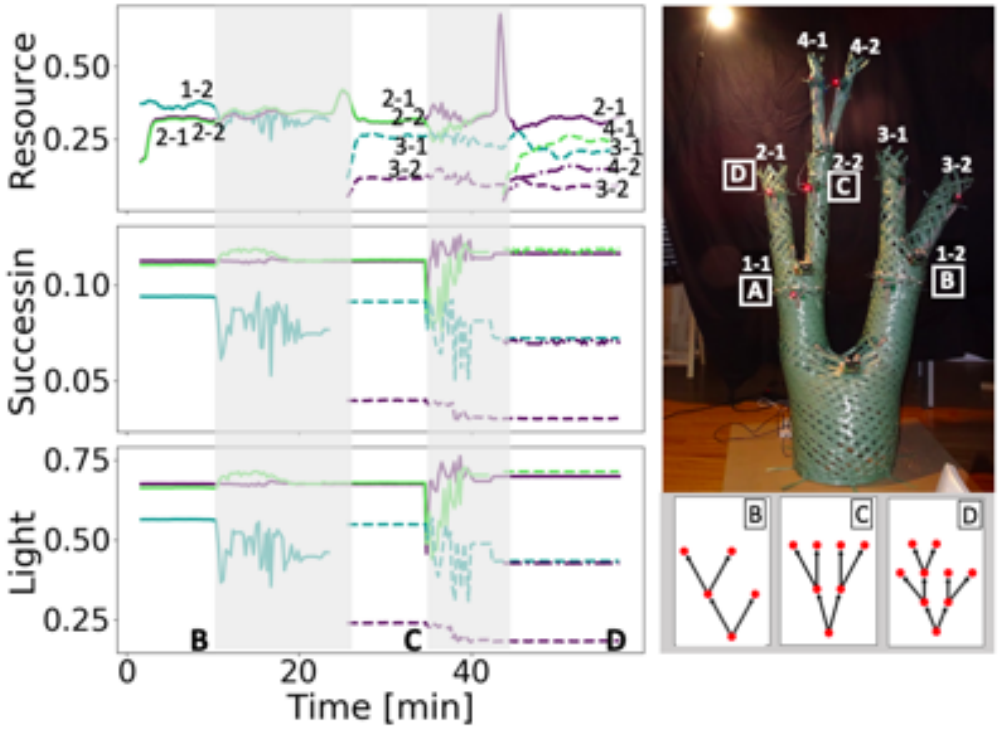


Figure 58: Growth with $\beta = 1$.The plots on the left demonstrate the values of resource, successin, and the light input for selected nodes over the course of the growth. The final structure is depicted on the right. The A-D labels in the plots mark the steps right before the start of manual growth. In the photo of the final structure, the labels indicate the position of growth at each step. The shaded parts of the plots indicate the periods when growth was physically realized. The labels 1-1, 1-2, etc. mark the location of the selected nodes on the structure and their corresponding values (resource, successin, light) on the plots. The graphs on the bottom-right depict the virtual VMC graphs running on the structure at each step of growth.

Table 1 shows the resources and the light values of all the leaves, with the maximum resource value of each setup represented in bold and the maximum light values represented in italic fonts. As shown in the table, there is a very low change in the light values of each leaf between the different setups. In all the setups, the light received by the leaf 2-2 is the highest. The leaves 1-2 and 4-1 are in the next position although 1-2 is slightly higher. The leaf 4-2 receives the least amount of light. With $\beta_c = 1$ and $\rho = 0.99$, ordering the leaves based on their resource values directly reflects the order of their light values as: 2-2, 1-2, 4-1, 4-2. With the same $\beta_c = 1$ and low $\rho = 0.74$, the order changes in favor of the shorter branch preferring 1-2 over the 2-2. For $\beta_c = 2, \rho = 0.99$, the combination of the difference in the light values and the length of the branches seem to act quite clearly in favor of the longer branches. In this case both 4-1 and 4-2 as longest branches get more resource than 2-2- and 1-2, while the 4-1 is preferred due to higher light value. For $\beta_c = 2, \rho = 0.74$, the combination is more complicated. The 2-2 which is a leaf with highest light and medium length gets the highest resource. But the next choice is the long 4-1 even with a slightly less light comparing 1-2. Overall, the experiment indicates a tendency for shorter paths with smaller transfer rate and a tendency for further growth at the already grown branches for larger competition rate. This is in line with the discussion in D2.3.

Table 1: Combined effect of competition and transfer rates.

ρ	β_c	State var.	1-2	2-2	4-1	4-2
0.99	1.0	resource	0.252	**0.268**	0.247	0.219
		Light	0.809	*0.857*	0.793	0.702
0.74	1.0	resource	**0.358**	0.279	0.189	0.168
		Light	0.806	*0.854*	0.791	0.699
0.99	2.0	resource	0.112	0.227	**0.364**	0.287
		Light	0.806	*0.849*	0.787	0.698
0.74	2.0	resource	0.231	**0.289**	0.263	0.205
		Light	0.804	*0.853*	0.791	0.698

Regulating growth in particular branches by using a sensor-dependent transfer rate. In this experiment, the effect of the sensor-dependent transfer rate is shown. The final structure of Fig. 57 is used in the ambient room light. After the first few minutes of the experiment with the intact structure, the leftmost branch is bent such that I_{tilt} decreases considerably. Fig. 59 shows the variable values over the course of the experiment. It shows that bending a branch leads to small values of I_{tilt}, decreases the transfer rate in the associated interior node and results in a lower share of resource for that branch which may eventually restrict its growth.

The effect of the adaptation rate. In this experiment, we investigate the effect of the adaptation rate on the speed of resource dynamics. A directional light source is placed at the top-left of the structure demonstrated in Fig. 60 (top). In order to investigate the response time of the system to the changes in the environmental input, an experimenter casts shadow on the leftmost branch of the structure in various intervals of time. The experiment is repeated with two different values of adaptation rate $\alpha \in \{0.1, 0.9\}$. The light intensity as perceived by the sensors, successin production, and resource values of all the three leaves of the structure are observed. Fig. 60 (bottom) demonstrates the variable values during the course of the experiment for both values of α. The figure shows slow changes in the resource level with the low value of $\alpha = 0.1$ that reflects a slow change in the vessels that act as a sort of memory of the system stored spatially. On the other hand, with $\alpha = 0.9$, resource values respond very quickly to the changes in the sensor inputs, reflecting a fast change in the thickness of vessels. Depending on the application, fast or slow response times can be desirable. For example, the slow dynamics of

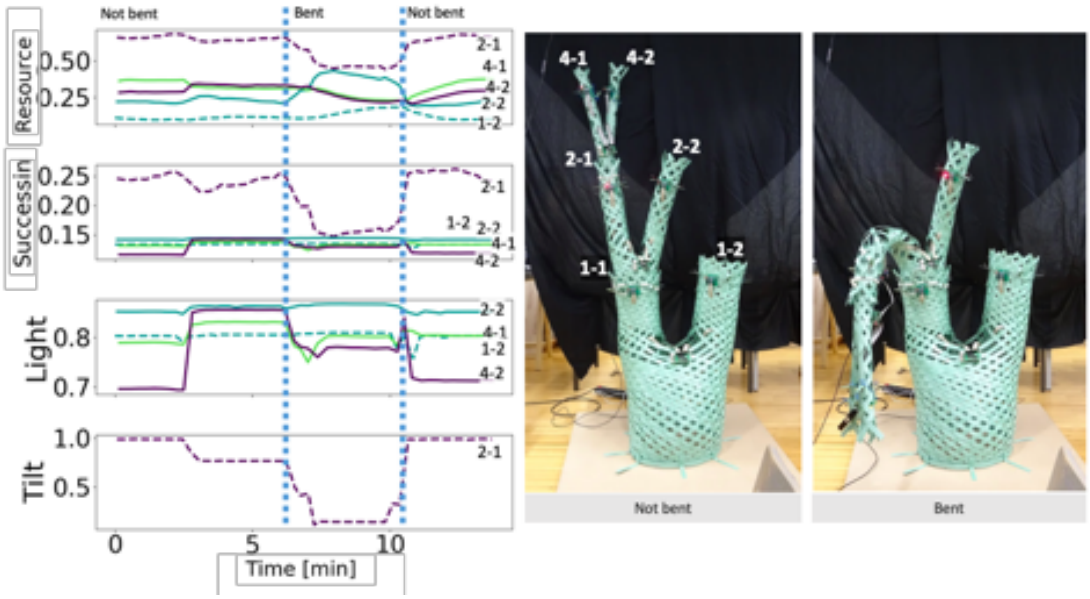


Figure 59: Values of resource, successin, input light variable (sensed at the leaves), and the tilt variable (sensed at the interior node) in the course of the experiment for a setting using the sensor-dependent (tilt-dependent) transfer rate. The structure is intact at first, then the left branch is bent for a period of time (indicated in the plots and depicted on the right figure), and then it is released again. The tilt value is shown only for the interior node 2-1 which is the node that senses the bending. The light input values are shown for the leaves. The resource and successin values are shown for both the leaves and the interior node 2-1.

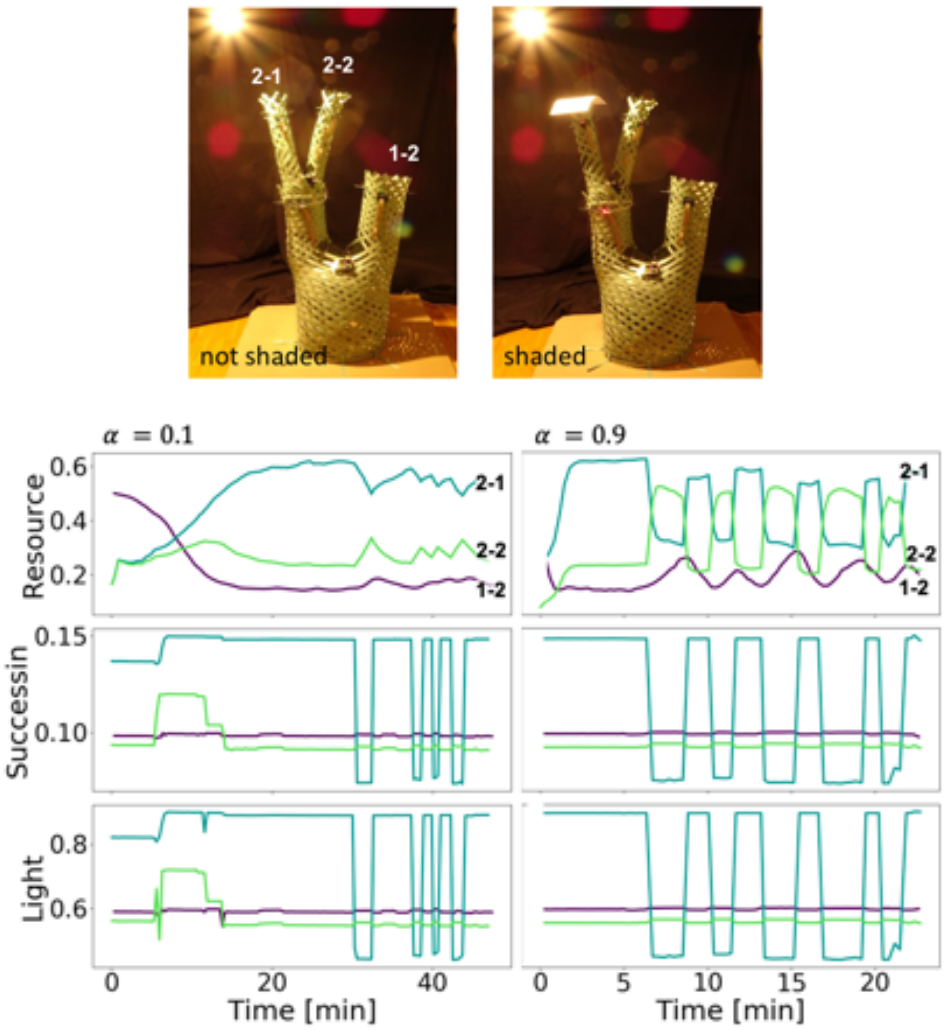


Figure 60: Top: the structure in both shaded and unshaded conditions. Bottom: the variables over the course of the experiment with periods of shaded and unshaded conditions with two different values of adaptation rate α. The results show a slow reaction to the change in lighting condition (shading/unshading) for small α and a quick change for the large α.

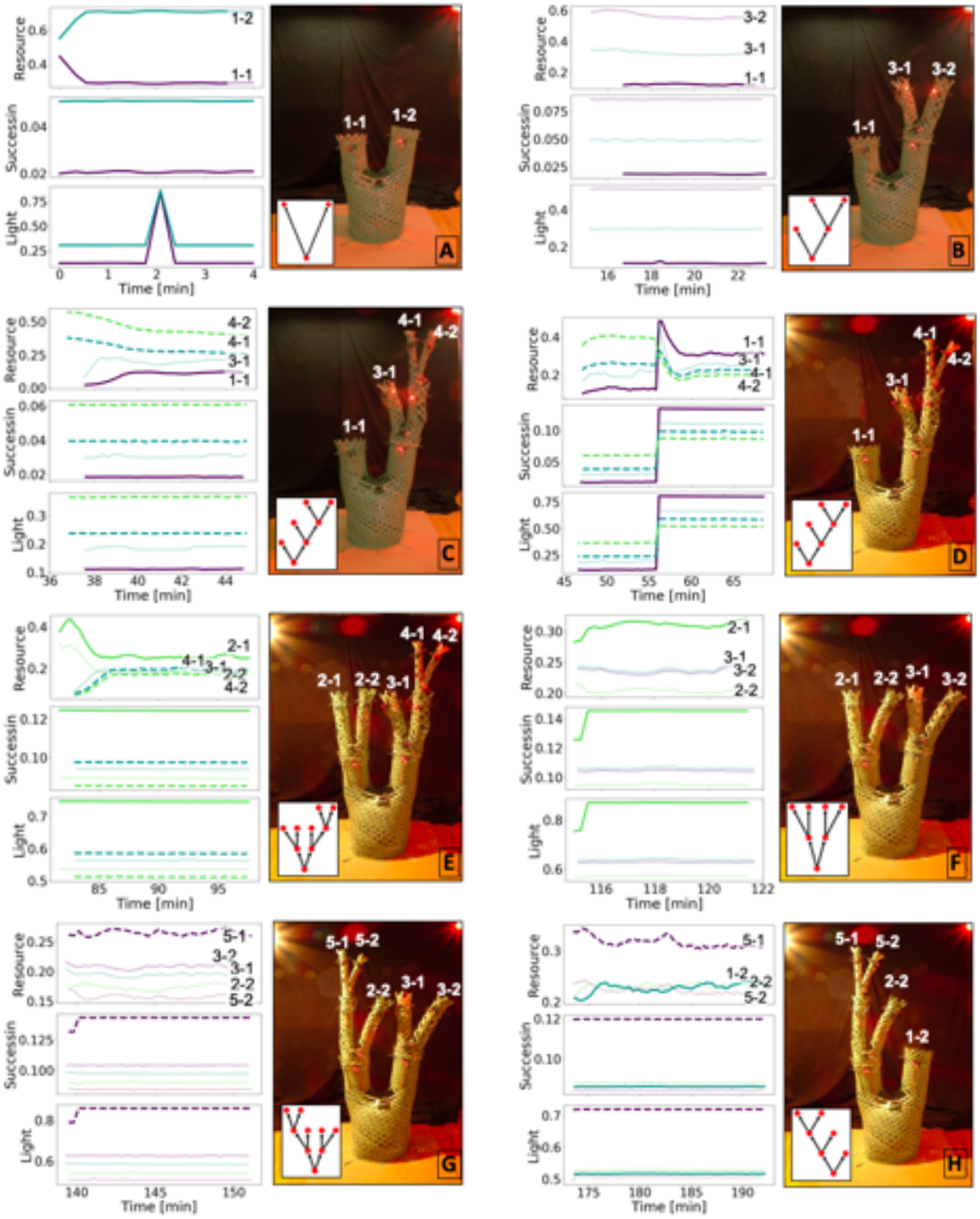


Figure 61: The shape of the structure changed due to changes in the environmental inputs. In the first stage (A-C) only a weak light is switched on at the top-right. After the growth of the structure towards the light, a stronger light is also switched on at the top-left (D). The shape of the structure changes accordingly (E-H).

the vessels leading to a slow response in the experiment with $\alpha = 0.1$ filters out the changes in the light as an environmental noise and prefers the leftmost branch as the candidate of further growth during the whole experiment, while the experiment with $\alpha = 0.9$ switches between the leftmost and the middle branch as the candidates for growth.

Adaptation of shape to changes in the environment Fig. 61 shows an example of adaptation of shape to changes in the environment. The value of $\beta = 1$ and both addition and deletion are possible. The experiment is performed in two stages. In the first stage, a weak light is switched on at the top-right of the structure. The structure starts from an initial module and grows towards the light (steps A-C in Fig. 61). In the second stage, a stronger light is added to the environment at the top-left of the structure (step D in the figure). The structure reacts to the change by growing new modules at the left side and losing the modules at the right side (steps E-H) such that the final structure is a reflected transformation about the y-axis of the structure at the end of the first stage. Fig. 61 shows the shape of the structure at each step of the growth process and the values of resource, successin and light at every leaf.

5.3 Large-scale adaptation and self-repair

The VMC structures built in braid above are relatively limited in scale. Here we augment the structures by 2D simulations exploring artificial growth by VMC at larger scales. In these simulations, the braid branches and braid intersections shown above can here be represented simply by nodes and connections in the tree structure. In these similations, the un-added braid modules that the user has the possibility to add to the structure in the setup described above, can here be represented by free-moving particles in the area around the tree structure. In these simulated experiments, we test large-scale artificial growth by VMC for the tasks of adaptive growth, collective site selection, and self-repair.

5.3.1. Implementation details for tree growth in simulation

Here we use our vascular morphogenesis model to grow adaptive and self-repairing tree structures according to light conditions in the environment. The seed node has a fixed ID of 0 and every other node uses a one-byte random ID and keeps it throughout the experiment. The birthday paradox among the nodes may seem inevitable (i.e., at least two nodes sharing the same ID), however we did not encounter noticeable drawbacks due to ID conflicts and our tests with unique IDs for each node show a negligible difference in swarm performance. The joining process used here is of higher complexity than that described above, as the tree structure does not necessarily accept a moving particle's request to join. All nodes follow a standard messaging protocol and 'narrowcast' to their neighborhood. The message is nine byte in size and includes: the control parameters (S, V, and R); the sum of the children's V; the ID; the listener's ID in the case of a direct communication channel; the number of children still; whether the node is looking for a parent; a one bit confirmation message for the listener to join as a child (which is dependent on several factors, explicitly, whether the listener is a moving particle and is a relevant candidate to join as a child, its current state, and a notification of completed joining with confirmation); a message announcing intention to leave the tree; and finally the light level.

Similar to the directed aggregation procedure, a joining particle has probability P_i to join the tree, in this case depending on the R available at the point of entry, given by

$$P_i = Pr\left[X < \frac{R_i}{R_{\text{root}}}\right] , \tag{1}$$

Algorithm 1: Pseudocode for adaptive self-assembly and self-repair

```
while true do
    measure light;
    send message;
    if in walking state then
        move randomly;
        if message received then
            if from tree and confirmed then
                stop;
                join tree;
            else if from walking or saturated agent then
                turn;
            end
        end
    else
        // in tree
        calculate parameters;
        if resource is below threshold then
            leave tree;
            reset;
        end
        if message received then
            if from potential child and not saturated then
                add child to children list;
            else if from child or parent then
                update parameters;
            end
        end
    end
end
```

where R_{root} is the seed resource. If the node is not saturated and $X < \frac{R_i}{R_{\text{root}}}$, then the node is able to get one more child. Accordingly, the seed node is always able to get saturated, while another node with low R might get any children.

A moving particle considers the content of the incoming message only if it comes from a node in the tree. Our implementation allows the particles to join and leave the tree structure at any time during the experiments. The possibility of nodes pushing each other is reduced by triggering low speeds in the case of close proximity to the tree.

When a node gains a new child, a direct communication channel is virtually established by incorporating the listener's ID in the message. After receiving the confirmation message from a potential parent node in the tree, the respective joining particle finalizes the process and joins the tree. The parent and child continually update their parameters as long as the connection is available. If this communication line breaks and either of them is not able to hear from the other one for a given period of time, the child node will trigger the leaving process. The parent node removes the child from its list and attracts moving particles again. The leaving node ignores incoming messages for given time period in order to leave the area. Leaving can also be triggered by a lack of R, when R is below the threshold for five time steps.

Asynchronous communication disrupts information flow through the branches of the tree, which is crucial in VMC for self-organized control of the growth process. Parameters need time

to traverse the tree structure and sudden changes demand further verifications to ensure that the tree does not collapse.

For instance, while adding a new child the weight V_i of a node i may lead to $V_i > V_{\text{all}}$, which means that the R received by the children should exceed that of the parent—however, this is impossible. In our implementation, we prevent these unstable conditions by giving a time buffer and applying some constraints when assigning R.

Another of our VMC modifications is specifically relevant for the site selection experiments, where the tree should be able to spread and explore, despite a lack of S in the environment. For this case, we allow an even distribution of R when none of the children can supply S to the tree. This allows the tree to expand in a barren environment, so that it can explore until it finds an S-rich environment somewhere else. Similar to natural plants, the tree structure continuously grows regardless of its state and size. Therefore, the tree regrows damaged portions, adapting to any environment changes in the meantime. The simplified algorithm used for our plant-inspired self-assembly is explained in Alg. 1.

5.3.2. Simulation results

Adaptation to a dynamic environment. In all 20 experiments with simulated nodes, the swarm succeeds in growing a tree in the correct direction, and then succeeds in adapting to the environment reversal by dissolving its now obsolete tree and growing a new one in the opposite direction.

The swarm reacts to changes in the light conditions in a selected experiment. Initially the swarm contains only a seed node, forming the root of the tree structure. The tree grows rightward toward the bright zone. The light conditions transition gradually, to the opposite configuration which is maintained for the experiment remainder. As a result of the light transition the swarm adapts itself, adding and removing nodes to the tree, keeping the majority of nodes exposed to brightness.

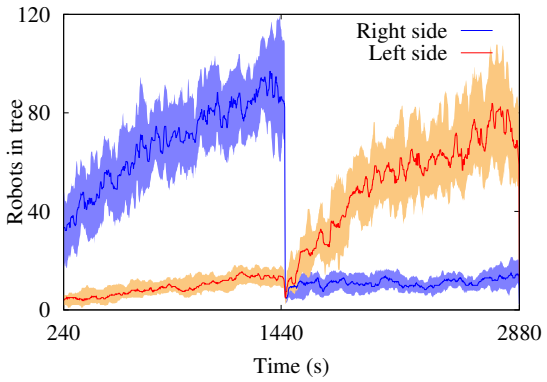


Figure 62: Median number of nodes in the tree structure over time during adaptation experiments. The median values of twenty runs are plotted.

The concentration of movement in all parts of the arena decreases over time, caused by the formation of a tree during that period. Movement concentration is substantially lower in the currently bright right-hand zone, where the tree is growing. The tree gradually disassembles, causing a noticeable jump in the red line associated with movement in the right-hand zone. The left-hand zone that is now bright contains the majority of the tree structure, and therefore

displaying a lower concentration of movement.

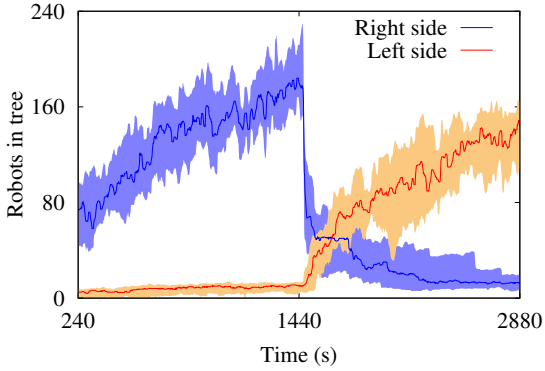


Figure 63: Median number of nodes in the tree structure over time during site selection experiments. The median values of twenty runs are plotted.

The results (see Fig. 62) show that in the beginning the number of nodes in the tree structure—located at the bright side, right-hand—is rising over time ($t < 1440$s). After the environment change, the tree dissolves from the right side and moves to the newly bright left-hand side. The results verify the capability of the swarm to adapt its self-assembled structure to changes in the environment.

Site selection. In all 20 experiments with simulated nodes (see Fig. 63), the swarm succeeds in finding and selecting the more advantageous site, and succeeds in adapting its choice after changes in the environment. During the first phase the rightward zone is closer to the seed than the leftward, making the left area harder to reach even though they are equally bright. Therefore, the swarm collectively decides build a tree structure rightward. In the second phase, the gap on each side of the seed is equal, but the rightward zone becomes less bright. As a result, the tree disassembles and rebuilds itself leftward. This shows that our control method not only succeeds in choosing the closest of two bright sites and in distinguishing between the quality of two equally near sites, but also is sensitive enough to balance the factors of quality and proximity and adapt its structure appropriately.

Self-repair of damage. In all 20 experiments with simulated nodes (see Figs. 65, 66, and 67), the swarm successfully regrows its tree structure after the majority of it is damaged. A dark bar is added in the bright area of the arena, to simulate damage by 'cutting' the structure self-assembled by the swarm. A structure forms before the dark bar is added. The tree successfully adapts itself to the bright area between the seed and the 'cutting' bar, and repairs itself after the barrier is lifted. There is increased motion after projecting the 'cutting' bar, showing the damage and recovery processes, until the motion declines again. In Fig. 64, the blue area demonstrates the median number of the nodes in the tree structure in twenty simulation runs. The tree size suddenly drops around $t = 1440$ when it is exposed to the 'cutting' bar. Soon after lifting the bar, the damaged area of the tree structure grows back, repairing the self-assemblage.

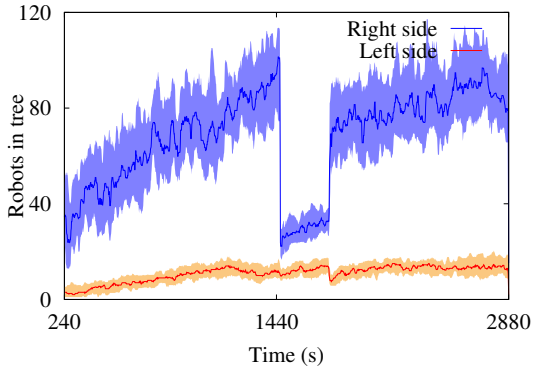
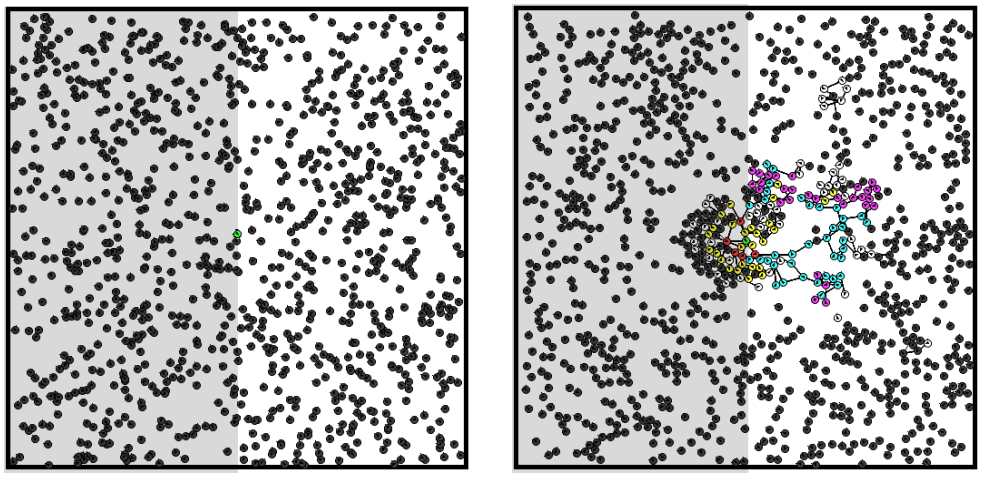
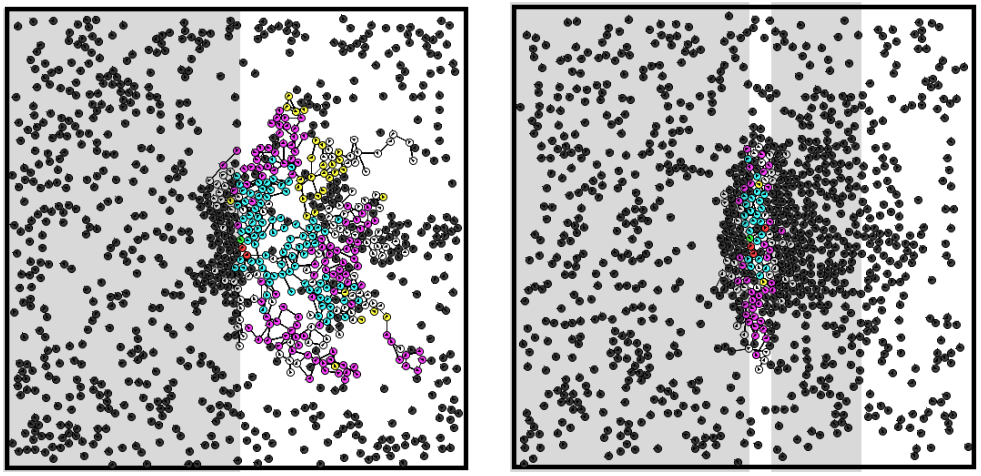


Figure 64: Median number of nodes in the tree structure over time during self-repair experiments. The median values of twenty runs are plotted.



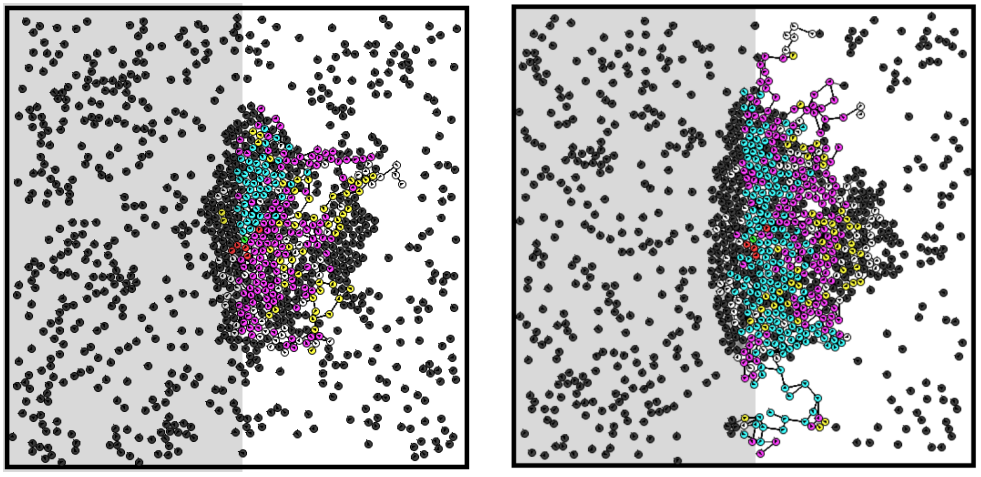
(a) $t = 0$ min (b) $t = 9.5$ min

Figure 65: Starting third of the self-repair experiment, showing the tree structure formed, every 9.5 minutes. Resource levels from high to low are green, red, cyan, purple, yellow, and white. Nodes in black are not part of the tree structure.



(a) $t = 19$ min (b) $t = 28.5$ min

Figure 66: Middle third of the self-repair experiment, showing the tree structure formed, every 9.5 minutes. Resource levels from high to low are green, red, cyan, purple, yellow, and white. Nodes in black are not part of the tree structure.



(a) $t = 38$ min (b) $t = 47.5$ min

Figure 67: Final third of the self-repair experiment, showing the tree structure formed, every 9.5 minutes. Resource levels from high to low are green, red, cyan, purple, yellow, and white. Nodes in black are not part of the tree structure.

6 Conclusions

This concludes deliverable *D3.3 flora robotica as Social Garden*. The deliverable has reported on the construction of a physical Social Garden site and the infrastructures developed to support various levels of communication and social interaction between plants, robots, braids and humans. Growth dynamics of the Social Garden have been demonstrated with a focus on artificial growth of braided scaffolds and self-repairing behaviour of the symbiont. We have also presented the cloud-based API for connecting the Social Garden to the internet, and reported on the development of virtual environments to support social and cultural practices of virtual exploration via interactive evolution that targets the investigation of braid morphologies, symbiont morphologies, and controllers. This deliverable documents that we have achieved the objectives of the work package task.

More generally, the objectives of work package 3 stated in the Description of Work are:

> This work package has three main objectives:
>
> 1. Develop appropriate methods of architectural representation of *flora robotica* and design rules to make it possible to integrate this technology into architectural practice;
> 2. Develop construction logics and propositions for the use of *flora robotica*;
> 3. Develop the social and cultural dimension of *flora robotica*.

Through a supplementary architectural design presented in this deliverable - that serves the primary purpose of speculating on the architectural possibilities of the Social Garden concept at larger scales - we have added further insight into the principle objectives of this work package with new design workflows, evaluated construction logics, spatial proposition, and development of a specific bio-hybrid aesthetic language. Although speculative, the proposition is grounded in the insights and technologies developed throughout the project, and hints at new social and cultural opportunities that our bio-hybrid architecture could promote if realized 'in the field.' We have therefore fulfilled the objectives we set out to achieve.

References

[1] Phil Ayres, Alison Grace Martin, and Mateusz Zwierzycki. Beyond the basket case: A principled approach to the modelling of Kagome weave patterns for the fabrication of interlaced lattice structures using straight strips. pages 72–93.

[2] Christopher R. Bonham and Ian C. Parmee. An investigation of exploration and exploitation within cluster oriented genetic algorithms (COGAs). In *Proc. of the 1st GECCO Conf.*, 1999.

[3] Jeff Clune and Hod Lipson. Evolving 3d objects with a generative encoding inspired by developmental biology. *ACM SIGEVOlution*, 5(4):2–12, 2011.

[4] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11–15, Pasadena, CA USA, 2008.

[5] Daniel Nicolas Hofstadler, Joshua Cherian Varughese, Stig Anton Nielsen, David Andres Leon, Phil Ayres, Payam Zahadat, and Thomas Schmickl. Artificial plants-vascular morphogenesis controller-guided growth of braided structures. *arXiv preprint arXiv:1804.06343*, 2018.

[6] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE trans. on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[7] William J. Lucas, Andrew Groover, Raffael Lichtenberger, Kaori Furuta, Shri-Ram Yadav, Ykä Helariutta, Xin-Qiang He, Hiroo Fukuda, Julie Kang, Siobhan M. Brady, John W. Patrick, John Sperry, Akiko Yoshida, Ana-Flor López-Millán, Michael A. Grusak, and Pradeep Kachroo. The Plant Vascular System: Evolution, Development and Functions. *Journal of Integrative Plant Biology*, 55(4):294–388, apr 2013. ISSN 16729072. doi: 10.1111/jipb.12041. URL http://dx.doi.org/10.1111/jipb.12041http://onlinelibrary.wiley.com/doi/10.1111/jipb.12041/abstract;jsessionid=6FD7F0F489AA1C571711FDB984CD79F4.f04t04http://f1000.com/718002940.

[8] Ian C. Parmee. Cluster oriented genetic algorithms (COGAs) for the identification of high performance regions of design spaces. In *First Int. Conf. EvCA*, volume 96, pages 66–75, 1996.

[9] C. Preisinger. Karamba3d. URL https://www.food4rhino.com/app/karamba3d.

[10] Mostapha Sadeghipour Roudsari, Michelle Pak, Adrian Smith, et al. Ladybug: a parametric environmental plugin for grasshopper to help designers create an environmentally-conscious design. In *Proceedings of the 13th international IBPSA conference*, 2013.

[11] Jimmy Secretan, Nicholas Beato, David B. D'Ambrosio, Adelein Rodriguez, Adam Campbell, and Kenneth O. Stanley. Picbreeder: evolving pictures collaboratively online. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 1759–1768. ACM, 2008.

[12] Kenneth O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

[13] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[14] D. Stasiuk. cocoon. URL http://www.bespokegeometry.com/2015/07/22/cocoon/.

[15] G. van Rossum. Python. URL https://www.python.org/.

[16] Payam Zahadat, Daniel Nicolas Hofstadler, and Thomas Schmickl. Vascular Morphogenesis Controller: A generative model for developing morphology of artificial structures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 163–170, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4920-8. doi: 10.1145/3071178.3071247. URL https://doi.acm.org/10.1145/3071178.3071247.

[17] Payam Zahadat, Daniel Nicolas Hofstadler, and Thomas Schmickl. Development of morphology based on resource distribution: Finding the shortest path in a maze by vascular morphogenesis controller. *Proceedings of the European Conference on Artificial Life*, 14:428–429, 2017. doi: 10.1162/ecal_a_0071_14. URL https://www.mitpressjournals.org/doi/abs/10.1162/ecal_a_0071_14http://cognet.mit.edu/proceed/10.7551/ecal_a_071.

[18] Payam Zahadat, Daniel Nicolas Hofstadler, and Thomas Schmickl. Morphogenesis as a Collective Decision of Agents Competing for Limited Resource: A Plants Approach. In Marco Dorigo, Mauro Birattari, Christian Blum, Anders L. Christensen, Andreagiovanni Reina, and Vito Trianni, editors, *Swarm Intelligence. ANTS 2018. Lecture Notes in Computer Science*, pages 84–96, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00533-7. doi: 10.1007/978-3-030-00533-7_7. URL https://link.springer.com/chapter/10.1007/978-3-030-00533-7{_}7https://doi.org/10.1007/978-3-030-00533-7{_}7.