Quantitative comparison of self-reconfiguration algorithms for 3D Catoms

Gilles Abdel Ahad[1], Nancy Awad[1], Benoit Piranda[1], Jad Bassil[1], Justin Werfel[2], and Julien Bourgeois[1]

[1] Université de Franche-Comté, FEMTO-ST Institute, CNRS, France
firstname.lastname@univ-fcomte.fr,
[2] Harvard University, USA
jkwerfel@seas.harvard.edu

Abstract. Modular self-reconfigurable robots can be used to build objects either by filling the space entirely with robots (densely packed) or by constructing a hollow internal structure (scaffolding) and coating the outside. Scaffolding offers many advantages for building an object versus a densely packed representation. However, the different approaches have never been compared. In this article, we define and evaluate metrics for comparing scaffolding algorithms and we test them on three existing scaffolding algorithms.

Keywords: Modular Robots, Self-reconfiguration, 3D Catoms, Scaffold, Programmable Matter

1 Introduction

Unlike traditional rigid-bodied robots, modular self-reconfigurable robot systems (MSRs) consist of individual modules that autonomously connect and disconnect, enabling them to assume multiple configurations. This ability, known as self-reconfiguration, grants MSRs the capacity to dynamically alter their shape and structure. The resulting adaptability and versatility empower MSRs to adeptly navigate various tasks, terrains, and environmental conditions. Additionally, their distributed and redundant architecture improves fault tolerance, allowing for continuing functioning even in the event of module failures or damage. The Programmable Matter Project [1] is a project within MSR research that aims to advance both hardware and algorithms for MSR technology. This ambitious initiative, growing out of the earlier Claytronics project [2], aims to harness the capabilities of MSRs to construct objects using micro-scale modular robots, termed Claytronics atoms (Catoms) [3]. One of the main developments made in this project is the *3D Catoms*, a novel MSR model, described in detail in Section 2.1, that will be the focus of our experiments.

Self-reconfiguration requires an algorithm to achieve a desired configuration, ideally as quickly and efficiently as possible, while taking into consideration multiple constraints. Target configurations can be divided into two classes: densely packed and hollow-shaped configurations. Densely packed configurations,

in which modules fill space without extraneous voids, are ideal for constructing rigid and robust forms. However, this approach comes with limitations, such as requiring a multitude of modules to fill the entire volume. Additionally, minimizing empty spaces within the structure hinders the overall maneuverability because only modules on the outer surface retain some ability to move. Both of these factors mean that MSRs using densely packed configurations will require more time to complete a reconfiguration operation. On the other hand, in hollow-shaped configurations, MSRs arrange their modules into a shape with internal voids or cavities, using fewer modules total and allowing motion of modules within the structure. This sparse framework upon which the configuration is built is often referred to as a scaffold. To achieve the final desired external form, a layer of "coating" modules is applied to the outer part of the scaffold. Three distinct scaffolding approaches have been proposed to facilitate the self-reconfiguration of *3D Catoms* [4–6]. For clarity and consistency throughout this paper, we will use the labels **Scaffold-1**, **Scaffold-2**, and **Scaffold-3** to refer to the respective works done by Thalamy et al. (2019) [4], Khawand et al. (2022) [5], and Bassil et al. (2022) [6]. Each of these three approaches offers a novel solution to the challenges of assembly and self-reconfiguration of *3D Catoms* for hollow-shaped configurations. A more detailed discussion of these three strategies is provided in Section 2.2.

In this article, we review and quantitatively compare the three scaffolding approaches {**Scaffold-1**, **Scaffold-2**, and **Scaffold-3**} for *3D Catom* reconfiguration, utilizing innovative experimental techniques to assess their performance. Section 2 presents the overall context of this work and offers a detailed discussion of the three strategies. Subsequently, Section 3 presents a series of experiments to compare and analyze them. Finally, we draw conclusions and outline future directions in Section 4.

2 Context

2.1 3D Catoms

MSRs consist of numerous individual units, each acting as an independent agent. These units have the capability to physically connect with one another and collaborate via communication to accomplish shared objectives. MSRs come in a variety of architectural forms characterized by how their modules connect. The most studied approach is the chain-type architecture, where modules link sequentially forming a tree-like structure, which among other attributes can simplify control of modules [7–9]. The second prevalent architecture is the lattice-type MSR, in which modules move between locations in a predefined ordered grid known as a lattice. This structured environment offers several key advantages. Lattice-type MSRs provide precise module localization since each module occupies a well-defined position within the lattice, adhering to the specific geometric constraints of the chosen lattice type [10, 11]. Various lattice structures exist or can be defined, each characterised by the arrangement of its constituent cells

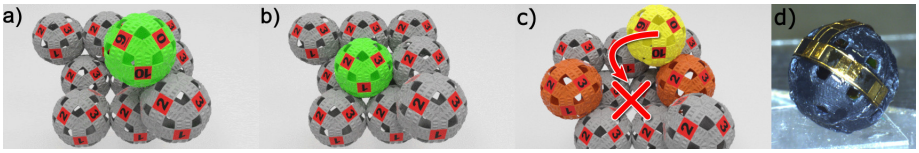


Fig. 1. The *3D Catom*: (a) and (b) show the movements that a *3D Catom* is capable of doing, (c) shows the bridging constraint and finally (d) shows a *3D Catom* with electrodes wrapped around the shell

also known as cell geometry, its density and dimensions. Additionally, lattice-type architectures offer greater flexibility in the forms that can be achieved during reconfiguration, compared to chain-type MSRs. Beyond these two prominent types, hybrid architectures have been developed combining elements from chain-type and lattice-type for potential flexibility gains [7]. Finally, free-form robots represent the other end of the spectrum, dynamically assembling into various configurations without a predetermined structure, offering maximum flexibility but also significant challenges in control and coordination [12].

In this paper, our focus is on *3D Catoms*, a lattice-based MSR, shown in Figure 1. *3D Catoms* [3, 13, 14] are millimeter-scale, quasi-spherical modules that use electrostatic attraction to connect and form various 3D structures. The physical design of *3D Catoms* addresses key challenges in connecting modules, enabling their movement within a 3D lattice, and facilitating communication between robots. The major constraint that *3D Catoms* can run into is known as the bridging constraint 1, which refers to the impossibility for a *3D Catom* to enter or leave a position that is surrounded by two opposing modules. In what follows we review three previously reported reconfiguration algorithms for *3D Catoms* using different scaffolding strategies.

2.2 Scaffolding strategies

Scaffolding strategies, first introduced in [15, 16], offer a strategic approach to facilitate self-reconfiguration of MSRs. Scaffolds offer a structured path for modules to move and connect, reducing the complexity of coordinating large-scale reconfigurations, and thus simplifying movement planning, reducing collisions and achieving faster reconfiguration.

Scaffold-1

This scaffolding strategy, presented in [4], arranges modules into higher-level fundamental components known as tiles (Fig. 2). Each tile acts as a structural unit with a central module called "tile root" that coordinates construction. Other modules, called "tile components", branch out from the root in multiple directions. These branches include two horizontal ones along the x and y axes, and one to four vertical branches (along the vectors $\overrightarrow{v} = \pm 0.5\overrightarrow{x} \pm 0.5\overrightarrow{y} + \sqrt{0.5}\overrightarrow{z}$) colored blue in Fig. 2). Each branch originates from the tile root and has a

specific length b consistent across all the tiles. Each tile rests on four "incident branches" (grey in Fig. 2), each originating from one of the four parent tiles below, collectively creating a square pyramid-shaped base for the tile. To manage the flow of incoming modules, special modules called "Entry Point Positions" modules (purple in Fig. 2) are positioned centrally on the incident branches. These modules act as decision points, directing incoming modules to their designated destinations. The destination can be either a specific location within the current tile, where the module can contribute to construction, or a higher tile in the scaffold. Modules move through the scaffold by going from tile to tile vertically. They can rotate on special modules to change directions through branches.

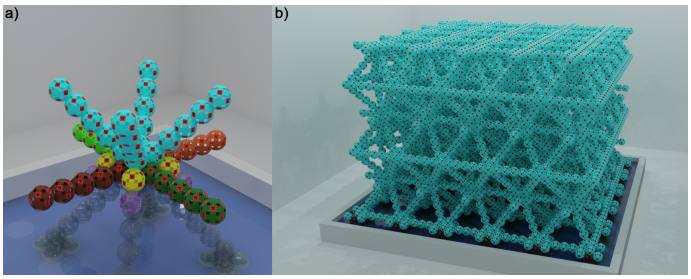


Fig. 2. Scaffold-1: (a) A single tile. Colors are defined in the text. (b) An object built using this scaffold.

Scaffold-2

This strategy, presented in [5], builds scaffolds using a different tile structure (Fig. 3). A tile is built using a leader module (yellow module on top of the black modules in Fig. 3) that acts as the conductor and coordinator of construction. There is also a controller module in each branch of a tile that once connected to the leader, it informs it, whether or not a tile must be continued. Upon receiving this signal, the leader uses the controller module's position as a reference to determine the target location for the next module. The leader then activates a new free module from the sandbox and guides it to its designated location on the branch. Once the tile is complete, the leader selects the topmost module on the vertical branch as the new leader for the next tile to be built on top. This process repeats, enabling the construction of multiple tiles stacked on top of each other. To parallelize the process, several neighboring tiles are built on the same horizontal level simultaneously. These neighbor tiles connect with each other by a common module while completing their horizontal branches, ensuring that the tiles connect smoothly without interlocking.

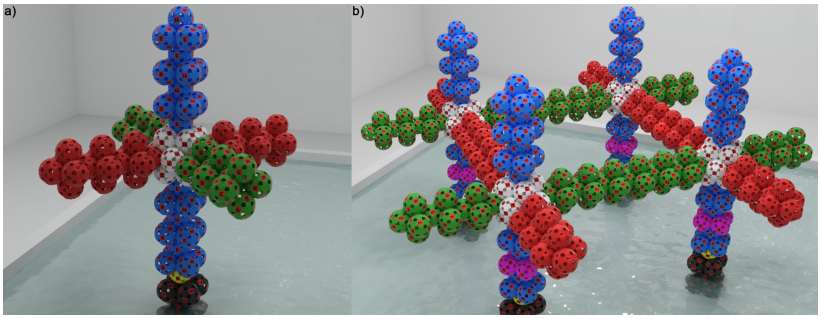


Fig. 3. Scaffold-2: (a) represents a single tile, (b) shows how multiple tiles connect to form a scaffold

Scaffold-3

The third scaffolding technique presented in [6], leverages the concept of "meta-modules" (Fig. 4).The concept is to group modules into logical units, each being a porous structure, facilitating the movement of free modules through the cavities. To define the structure of **Scaffold-3**, they propose meta-modules in two states, "Sparse" or "Full". A "Sparse" meta-module is composed of ten modules placed in a 3D hexagon-like shape in a face-centered cubic lattice. Each "Sparse" meta-module can fit in its internal empty volume a set of ten additional modules. A "Full" meta-module is a meta-module storing in its internal volume ten modules, i.e., constituting a total of twenty *3D Catoms*. These meta-modules present a significant advantage: creating a reserve of material or a storage place for modules. This allows to reduce the distances covered by the modules during the self-reconfiguration process, by providing a place to store incoming modules and/or providing a supply of outgoing modules close to their destination.

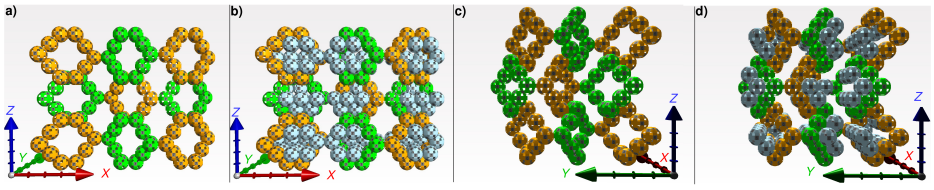


Fig. 4. Scaffold-3: (a) and (c) are "Sparse" meta-modules, (b) and (d) are "Full" meta-modules

2.3 Reconfiguration algorithms

Before describing the self-reconfiguration algorithms for the three scaffolding techniques, we note the importance of the assumptions each technique makes

about the initial locations of the modules. **Scaffold-1** and **Scaffold-2** assume a "sandbox", a virtual reservoir from which modules emerge to participate in constructing a target shape. **Scaffold-3** explicitly considers the positions of all modules at all times, so that a detailed initial configuration must be specified. These differences arise from the first two strategies being focused on *self-assembly* and the third on *self-reconfiguration*.

Reconfiguration through Scaffold-1
Two algorithms have been proposed using **Scaffold-1**, the first being a synchronous method [17] and the second an improvement with an asynchronous distributed algorithm [4]. In our study, we specifically investigate the second asynchronous algorithm. The proposed algorithm assumes the existence of a sandbox, a repository for modules, which allows modules to be introduced at various locations within the reconfiguration space and bring the initial communication that provides the description of the goal shape. The process involves two levels of planning. Firstly, a "Coordinator" module primarily guides the construction of a single tile by directing incoming modules to fill a component of that tile or to reach one of the children tiles. Secondly,"FreeAgent" modules exit the scaffold and navigate the structure using built-in rules while communicating with "Beam" modules to avoid collisions. "Beam" modules facilitate message forwarding between neighboring modules and regulate module flows to prevent collisions and deadlocks by granting permission for "FreeAgents" to move, according to certain rules.

Reconfiguration through Scaffold-2
The algorithm described in [5] helps create different 3D shapes using interconnected tiles within the **Scaffold-2** framework. This algorithm takes as input the desired final shape and the unified size of the branches of a tile, which determines the area occupied by a tile and consequently dictates the required distance between the sandboxes to create tiles with the ability to correctly connect their branches. The algorithm begins with multiple sandboxes, each responsible for providing modules to vertically construct tiles, layer by layer. Once a tile is built, as described in the previous section 2.2, the algorithm checks if the desired height for the target structure in the corresponding position of the sandbox has been reached. If so, construction stops at that level. If not, the algorithm continues adding tiles vertically, while also connecting the branches of neighboring tiles in parallel, to construct the scaffold and achieve the desired final shape.

Reconfiguration through Scaffold-3
This work differs from previous reconfiguration algorithms designed for **Scaffold-1** and **Scaffold-2**. Those algorithms focus solely on assembling a shape from modules stored beneath it in a sandbox. In contrast, for **Scaffold-3**, two self-reconfiguration algorithms have been proposed: a synchronous algorithm [18] and a hybrid centralized/distributed self-reconfiguration algorithm [19]. Both approaches aim to transform an initial shape into a desired configuration. Our

experiments specifically concentrate on the latter algorithm. This algorithm focuses on transforming an initial configuration (I) into a goal configuration (G). First, it classifies meta-modules into three categories: present only in I (denoted $I\backslash G$), only in G (denoted $G\backslash I$), or in both. Meta-modules falling only under I require dismantling, with their constituent modules redistributed within the structure to occupy vacancies within $G\backslash I$, forming new meta-modules as necessary. Using a Max-Flow algorithm, a centralized global planner identifies flow paths for modules from $I\backslash G$ to $G\backslash I$. Then, it employs a traffic-light-like system for distributed flow control of modules avoiding collisions and deadlocks.

3 Conducted experiments

In this section, we compare the three scaffold structures described above as well as their respective reconfiguration algorithms, to assess their strengths and weaknesses. All of the experiments are done using the VisibleSim simulator [20, 21], which was also employed for implementing the three works under consideration. VisibleSim is an optimized C++ framework for building lattice modular robot simulators, the simulations are controlled by distributed programming.

3.1 Comparison criteria

To effectively compare the various techniques of self-reconfiguration through scaffolding, we establish specific criteria to evaluate the performance and limitations of both the structures themselves and the reconfiguration algorithms used. Here is a breakdown of the specific criteria used for comparison:

- **Resolution of a scaffold:** reflects the granularity of each tile's resolution, affecting how precisely construction can be managed and what can be built.
- **Filling ratio:** is the ratio of the volume occupied by the robots to the total volume of the structure, impacting both material efficiency and structural properties.
- **Communication robustness:** assesses a structure's ability to withstand connector failures by having redundancies within the system.
- **Speed of construction:** measures how quickly specific structures can be built.
- **Robustness to errors:** evaluates how well the system handles errors caused by unforeseen factors.

3.2 Resolution of each structure

The tiles on each structure have different sizes which will result in slightly different size objects. The accuracy is determined by the size of each tile for both **Scaffold-2** and **Scaffold-3**, while for **Scaffold-1** it can be more accurate. In fact, **Scaffold-1** is able to increase the size of the object by 1 and reduce it by 1 from any side; it works by checking whether or not a certain *3D Catom*

is inside the targeted object before adding it, so that it has a resolution of 1. **Scaffold-2** uses a tile of size $8 \times 8 \times 8$ modules, including sandbox tiles of height 5, which allows us to conclude that its resolution is 8 *3D Catoms*. **Scaffold-3** uses meta-modules of size $3 \times 2 \times 4$, which makes its resolution different for every axis: 3 for the x axis, 2 for the y axis and 4 for the z axis.

In the remaining experiments, we construct two cube configurations with varying side sizes to compare the three scaffolding approaches and their corresponding algorithms. It's important to note that due to the resolution limitations of each tile, achieving perfect cubes with exact side sizes is not feasible for all three approaches. The first configuration is for a cube with an approximate side size of 10 *3D Catoms*, and the second configuration has an approximate side size of 20 *3D Catoms* 5.

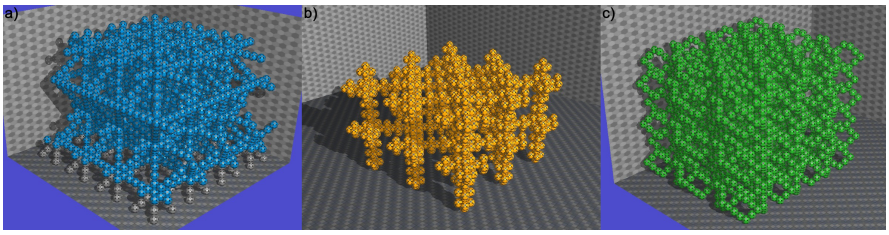


Fig. 5. All cubes of approximate side size of 20: (a) **Scaffold-1**, (b) **Scaffold-2**, (c) **Scaffold-3**

3.3 Filling Ratio

Sparse scaffolds reduce the number of modules needed to form a given shape, and facilitate the movement of modules for self-reconfiguration. However, having fewer modules may bring disadvantages like less structural rigidity. Hence we next compare the three approaches with respect to their filling ratio *FR*, defined as the number of *3D Catoms* required by a scaffold for the creation of an object (N_{object}) divided by the number of *3D Catoms* that would be required if the same object were completely filled with a solid mass of modules (N_{dense}). This particular measure is useful for comparing the scaffold structures without comparing the algorithms that are used for the creation of the objects. We compare the filling ratio for two cube configurations of approximate side lengths 10 and 20. Additionally, we compute the filling ratio for single tiles, to compare this measure for the three approaches in the limit of large structures.

All three sparse scaffold structures have FR values of $\sim$10 –-15 % (Table 1) for the cubes of side lengths $\sim$10 and $\sim$20. The scaffold that ends up using the most modules is **Scaffold-3** because of the size of each meta-module which is predefined to contain 10 modules and have a height of 4. It allows more precision for smaller objects but from the cube of approximate side size 10 to side size

	Scaffold-1	**Scaffold-2**	**Scaffold-3**
Cube size ($l \times w \times h$)	$10 \times 10 \times 10$	$11 \times 12 \times 9$	$11 \times 12 \times 8$
N_{object}	186	235	240
N_{dense}	1,668	1,884	1,728
Filling ratio (FR)	0.112	0.125	0.139
Construction time (s)	66,401	27,216	32,701
# motions processed	899	1,960	1,740
Cube size ($l \times w \times h$)	$20 \times 20 \times 20$	$18 \times 20 \times 17$	$19 \times 20 \times 20$
N_{object}	1,374	1,209	1,750
N_{dense}	12,215	9,108	11,620
Filling ratio (FR)	0.112	0.133	0.151
Construction time (s)	148,314	56,032	82,855
# motions processed	11,240	15,158	24,740
Repeating pattern (large-structure limit)	$11 \times 11 \times 6$	$9 \times 9 \times 9$	$8 \times 8 \times 5$
N_{object}	35	47	40
N_{dense}	726	729	320
Filling ratio (FR)	0.048	0.067	0.125

Table 1. Quantitative comparisons for the three approaches: filling ratio, speed of construction, # motions for reconfiguration. Construction time and # motions are not given for the large-structure limit since both increase without bound in that limit.

20 the filling ratio increases the most compared to the others. **Scaffold-2** has a better filling ratio than **Scaffold-3** for both cubes which can be explained by the size of each tile which is 8 modules high. And finally thanks to its very high resolution and the shape of each of its tiles **Scaffold-1** performs the best overall.

For structures that are very large in all three dimensions, the filling ratio for the full structure would be dominated by the value for a repeating pattern which happens to be a single tiles for **Scaffold-1** and **Scaffold-2**. Table 1 also reports FR for the three approaches for this large-structure limit. **Scaffold-1** and **Scaffold-2** have tiles that are parameterized by branch size b, where larger b means lower density (lower FR) but also lower mechanical stability. For those two approaches, Table 1 gives FR for the value of b recommended by the original papers ($b = 6$ and $b = 2$, respectively) [4,5]. However, if a structure could mechanically support it (e.g., slowly reconfiguring structures in zero gravity), FR can be made arbitrarily small as b becomes arbitrarily large.

3.4 Communication Robustness

3D Catoms use electrostatic electrodes to enable the connections between neighboring modules. Those electrostatic electrodes create single connections that can

be broken. The use of lattice structures helps avoid modules becoming completely disconnected in case of such failures. Despite these structures' robustness to such issues disconnections can still happen. In this section we test the different scaffold structures' communication robustness to determine how overall connectivity is affected as communication failures become more frequent.

For such a test, we need to simulate broken connections between the modules within the final object. To simulate this type of error, we use cubes of approximate side size 20, and define an algorithm that breaks interfaces used to connect the modules together randomly. Using a percentage, we can adjust how many interfaces connecting modules get broken, allowing us to test for multiple amounts of broken interfaces. Finally, we elect a leader within the structure, to keep the comparison as fair as possible we choose a module connected to 4 neighbors at the center of the top of the object. The leader will broadcast a message to all connected modules, which in turn, relay it to their neighboring modules, creating a chain reaction until every connected module has received the message. This broadcast message allows us to count every module that receives the message and return the final value. We chose to run steps of 5 % increase for breaking connected interfaces, with 100 iterations per value.

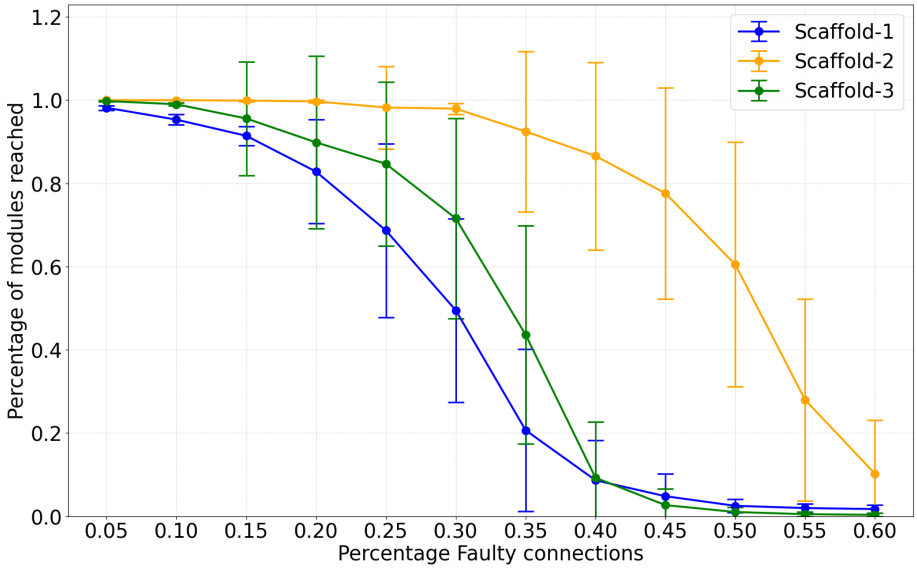


Fig. 6. Comparison of the robustness of different scaffold structures in the presence of broken connections. The graph shows the percentage of modules successfully receiving a broadcast message as the percentage of broken interfaces increases in 5% increments.

As we can see in Figure 6, **Scaffold-2** performs the best in this experiment; this can be explained by the 1 by 4 repeating pattern which reduces the chance of a full branch disconnection. The results for **Scaffold-1** are worse than **Scaffold-3**'s results. However, it is worth noting that **Scaffold-3** exhibits a notably larger standard deviation compared to **Scaffold-1**.

In addition to defining how many modules are getting disconnected it is important to note when disconnections start happening, and differentiate between disconnections of singular modules and multiple modules. The disconnection of a singular module can be less damaging than the disconnection of a group of modules because the latter usually affect a bigger area.

In **Scaffold-1**, the minimum connections one module can have is 1, hence a disconnection of a singular or multiple modules can happen from one broken connection because of the single connection lines of modules. In **Scaffold-2**, the minimum connections one module has is 4, hence a disconnection of a single module or multiple modules can start happening at 4 broken connections. In **Scaffold-3** the minimum connections a module has is 2, hence a disconnection of a single module or multiple modules can start happening at 2 broken connections.

If we look at the final object as a graph with *3D Catoms* as nodes and their connections to one another as the edges, then we can quantify the fragmentation of the structure in terms of the number of separate connected components of the graph. Figure 7 shows the average and the standard deviation of disconnected graphs in each cube of approximate side size 20 for each scaffold structure after 100 iterations of the algorithm. **Scaffold-1** shows an almost linear growth of the amount of disconnected graphs in relation to the percentage of failed connections, while starting at 40 ± 5 disconnected graphs for 5 % broken connections. **Scaffold-2** starts at 1 ± 0.5 disconnected graphs for 5 % broken connections, and keeps performing better than the other scaffolds through the rest of this experiment. **Scaffold-3** starts at 12 ± 3 disconnected graphs for 5 % broken connections, and performs better than **Scaffold-1** until 20 % broken connections.

3.5 Speed of Construction

Sparse scaffold structures enable modules' movement inside the structure during reconfiguration. Each of the tested three approaches has a different speed in constructing the final object depending on the motions of each module in the structure and on the reconfiguration algorithm it is using.

All three structures require a base layer of the scaffold at the bottom of the object, it is called a sandbox in the case of **Scaffold-1** and **Scaffold-2**'s algorithms. This sandbox allows to mimic a box of *3D Catoms* from which the new modules are added. For **Scaffold-3** we will be using the ASAPs algorithm [19] to have the newest implementation of this approach, and it will be creating the object through reconfiguration of an upside down version of that same object; this makes the process longer since the Catoms have to travel a longer route to arrive to the final position instead of coming from a sandbox like the other 2 algorithms.

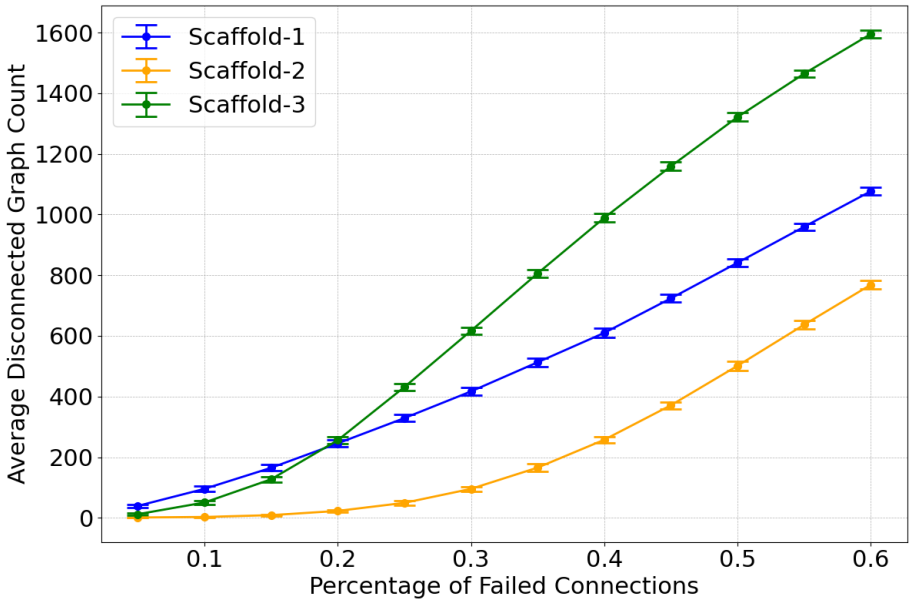


Fig. 7. Analysis of the fragmentation in the different scaffold structures by measuring the number of disconnected graphs as the percentage of broken connections increases. The graph shows both the average and the standard deviation of disconnected components in each scaffold structure after 100 iterations.

We should also mention that for all 3 structures the reconfiguration complexity is $O(h)$ with h being the final height of the structure.

We initially evaluated the speed of construction for a cube of approximate side size of 10 *3D Catoms* for all three algorithms. The results, presented in table 1, indicate that the algorithm for **Scaffold-2** significantly outperforms **Scaffold-3** , with **Scaffold-1** being the slowest. Subsequently, we conducted similar tests on a larger cube of approximate side size of 20 *3D Catoms*. As shown in table 1, scaling the object did not change the ranking we had previously, but **Scaffold-1**, **Scaffold-2** and **Scaffold-3** need 2.2, 2 and 2.5 times more time to finish this object respectively than the previous one. A first look at these results allows us to conclude that **Scaffold-2**'s algorithm and scaffold structure combination is the fastest. This result can be explained by the lack of a traffic light system that makes *3D Catoms* wait for a confirmation message from another *3D Catom* before moving to its next position, this systems adds safety checks to the system in case of failures.

Another way to compare the speed of construction would be the number of motions processed. This metric records each movement a *3D Catom* makes until the final configuration is reached. It allows a more fair comparison between the

speed of construction of all 3 structures since this doesn't take into account the traffic light system implemented in 2 out of 3 of the algorithms used.

The number of motions processed proves that **Scaffold-1** is the most efficient in terms of reducing the movement done by each *3D Catom* to arrive to its final position. **Scaffold-3** scores better than **Scaffold-2** for the cube of approximate side size 10 but proves to scale worse; this can be explained by the difference in size between the tiles of **Scaffold-2** and the meta-modules of **Scaffold-3**.

3.6 Robustness to Errors

There are different types of errors that could happen in practice during the building phase that challenge the reconfiguration algorithms.

In the case of a module failure that results in it being fully immobilized while the object is being built, the traffic light system of **Scaffold-1** and **Scaffold-3** would prevent collisions of other modules with the stalled one, allowing reconfiguration to continue other than for any parts of the structure that rely directly on the module that stopped. In **Scaffold-2**, the algorithm has no way of knowing a module stopped moving, and collisions between modules could occur—potentially a more serious problem for the system, for instance leading to cascading failures.

An example of an error that is probable is a drop in voltage on the connecting electrodes, this could cause the speed of movement of a module to slow down. If the reconfiguration algorithms contain a traffic light system such as **Scaffold-1** and **Scaffold-3** this slow movement would not affect the final object since the module wait their turn to move to their assign position. But, in case of **Scaffold-2** where there is no traffic light system the modules are sent to their position by the leader using synchronised movements, in case of such an error the margin of error would be equal to the time it would take for the next *3D Catom* to reach the slow module. This margin can be increased at the cost of speed of construction which is not ideal.

Since *3D Catoms* use electrodes for connections and movement, they will always be prone to having failures. If we take it that success of a full reconfiguration process requires every module movement along the way to be successful, then the number of motions n it takes to create the object (Tables 1, 1 for our examples above) relates the probability of success of a single module's movement P to that of success of the full reconfiguration P^n. For instance, suppose that individual motions are extremely reliable and P =99.99 % then the probability of success of building a cube of size 20 with **Scaffold-1**, **Scaffold-2** and **Scaffold-3** respectively is 32.5 %, 22.0 %, and 8.4 %. This observation suggests two conclusions. First, the number of motions required by an approach can have a very large impact on its probability of success in practice. More importantly, however, the success of building even a relatively small object can be low for any of the approaches, even if individual modules can be made to be extremely reliable. Therefore a high priority for future algorithm is to design approaches that explicitly accommodate failures of individual modules.

4 Conclusion

Every scaffold and algorithm studied above, proved to have its advantages and disadvantages as highlighted by the conducted experiments. For **Scaffold-3** while it did perform quite well in communication robustness and robustness to errors, it showed that its performance in speed and filling ratio would diminish with scaling. This ultimately is due to the fact that the algorithm while robust at dealing with errors is not designed for self-assembly but for self-reconfiguration. However, this is an advantage as it is the only algorithm between all three that permits going from an existing configuration to a targeted one. The architecture of the scaffold is convenient since it can store modules inside the structure thus widening the scope of what is reconfigurable. Similarly, **Scaffold-2** performs impressively well especially in communication robustness, with the extra connections per module. But the proposed algorithm can be criticized for its susceptibility to errors. Finally, **Scaffold-1** performs well for most criteria with high accuracy in reconfiguration, low total number of motions, and an impressive filling ratio, but it does seem to lack communication robustness mostly because of the minimal amount of redundancies compared to the other two scaffold structures.

For future work, comparison can be taken further and improvements can be made on each scaffold structure and their respective algorithms to enhance and apply them in different scenarios where their particular strengths can be greatly highlighted. Experiments should cover mechanical stress test of each structure to determine how they would react to stresses applied to the structure and compare their rigidity. Improvements can also be made by learning from each scaffold's strengths. For reconfiguration, the work presented for **Scaffold-3** is a step forward compared to the other two algorithms, enabling reconfiguration from initial configurations to a targeted one. Extending the other structures' algorithms to support self-reconfiguration outside a sandbox, would increase their versatility. To improve robust communication, structures should raise the redundancy of the system, enhancing its resilience against connection failures, as proved by the framework of **Scaffold-2**. In addition, a resolution of size 1 for the tiles should be considered for all future structures to create accurate size objects, as done in **Scaffold-1**. Finally, it is important to note that even while building small objects and having an elevated success rate of every motion the probability of finishing the object is rather low, hence the importance of accommodating failures of individual modules for future algorithms.

References

1. Julien Bourgeois, Benoit Piranda, André Naz, Nicolas Boillot, Hakim Mabed, Dominique Dhoutaut, Thadeu Tucci, and Hicham Lakhlef. Programmable matter as a cyber-physical conjugation. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002942–002947, 2016.

2. Seth Copen Goldstein and Todd C. Mowry. Claytronics: An instance of programmable matter. In *Wild and Crazy Ideas Session of ASPLOS*, Boston, MA, October 2004.
3. Benoit Piranda and Julien Bourgeois. Geometrical study of a quasi-spherical module for building programmable matter. In *Distributed Autonomous Robotic Systems: The 13th International Symposium*, pages 387–400. Springer, 2018.
4. Pierre Thalamy, Benoit Piranda, Frédéric Lassabe, and Julien Bourgeois. Scaffold-based asynchronous distributed self-reconfiguration by continuous module flow. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4840–4846. IEEE, 2019.
5. Carlita Khawand, Benoit Piranda, and Julien Bourgeois. A new scaffold for programmable matter objects. Technical report, Université de Franche-Comté, 2022.
6. Jad Bassil, Benoît Piranda, Abdallah Makhoul, and Julien Bourgeois. A new porous structure for modular robots. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1539–1541, 2022.
7. Behnam Salemi, Mark Moll, and Wei-Min Shen. Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3636–3641. IEEE, 2006.
8. Andres Castano, Alberto Behar, and Peter M Will. The conro modules for reconfigurable robots. *IEEE/ASME transactions on mechatronics*, 7(4):403–409, 2002.
9. Chao Liu, Qian Lin, Hyun Kim, and Mark Yim. Smores-ep, a modular robot with parallel self-assembly. *Autonomous Robots*, 47(2):211–228, 2023.
10. Satoshi Murata, Haruhisa Kurokawa, Eiichi Yoshida, Kohji Tomita, and Shigeru Kokaji. A 3-d self-reconfigurable structure. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 432–439. IEEE, 1998.
11. Morten Winkler Jorgensen, Esben Hallundbk Ostergaard, and Henrik Hautop Lund. Modular atron: Modules for a self-reconfigurable robot. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 2, pages 2068–2073. Ieee, 2004.
12. Petras Swissler and Michael Rubenstein. Fireant: A modular robot with full-body continuous docks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6812–6817. IEEE, 2018.
13. 3D-Catoms for Programmable Matter project. https://www.programmable-matter.com/technology/3d-catoms. Accessed: 2024-03-22.
14. Benoit Piranda and Julien Bourgeois. Designing a quasi-spherical module for a huge modular robot to create programmable matter. *Autonomous Robots*, 42(8):1619–1633, 2018.
15. Cem Unsal and Pradeep K Khosla. A multi-layered planner for self-reconfiguration of a uniform group of i-cube modules. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 598–605. IEEE, 2001.
16. Kasper Stoy and Radhika Nagpal. Self-reconfiguration using directed growth. In *Distributed autonomous robotic systems 6*, pages 3–12. Springer, 2007.
17. Pierre Thalamy, Benoit Piranda, and Julien Bourgeois. Distributed self-reconfiguration using a deterministic autonomous scaffolding structure. HAL https://hal.science/hal-02380222/file/25a15602-a6a9-451f-83e6-180d53c4f626-author.pdf, 2019. Accessed: 22/03/2024.

18. Jad Bassil, Benoît Piranda, Abdallah Makhoul, and Julien Bourgeois. Repost: Distributed self-reconfiguration algorithm for modular robots based on porous structure. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12651–12658. IEEE, 2022.
19. Jad Bassil, Benoît Piranda, Abdallah Makhoul, and Julien Bourgeois. Asaps: Asynchronous hybrid self-reconfiguration algorithm for porous modular robotic structures. 2023.
20. Benoit Piranda. Visiblesim: Your simulator for programmable matter. In *Dagstuhl Seminar*, 2016.
21. Pierre Thalamy, Benoît Piranda, André Naz, and Julien Bourgeois. Visiblesim: A behavioral simulation framework for lattice modular robots. *Robotics and Autonomous Systems*, 147:103913, 2022.