ijrr

The International Journal of
Robotics Research
00(000) 1–19
© The Author(s) 2011
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364910398160
ijr.sagepub.com
SAGE

On-line assembly planning for stochastically reconfigurable systems

Michael T Tolley and Hod Lipson

Abstract

Stochastic assembly approaches can reduce the power, computation, and/or actuation demands on assembly systems by taking advantage of probabilistic processes. At the same time, however, they relinquish the efficiency and predictability of deterministic alternatives. This makes planning error-free assembly sequences challenging, particularly in the face of changing environmental conditions or goals. Here we address these challenges with an on-line approach to assembly planning for stochastically reconfigurable systems where the spatial and temporal availability of modules is uncertain, either due to a stochastic assembly mechanism, resource fluctuations, or large numbers of uncoordinated agents. We propose an assembly algorithm that is guaranteed to find an assembly path for finite-sized, connected objects. This is achieved by sampling the space of possible assembly paths to the target structure that satisfy assembly constraints. Assembly is accelerated by pursuing multiple paths in parallel. The algorithm computes these parallel assembly paths on-line during assembly and is thus able to adapt to changing conditions, as well as predict the remaining assembly time. For situations where the number of paths found exceeds the number that can be pursued in parallel, the assembly algorithm further maximizes assembly rates according to domain-specific local assembly costs.

Keywords

Cellular and modular robots, assembly planning, self-assembly, biologically-inspired robots, micro/nano robots, reconfigurable robots

1. Introduction

The scaling of modular robotics and other reconfigurable systems to large numbers of small-scale units is reaching its limits due to a variety of technical challenges (Yim et al., 2007). The high power, connectivity, and actuation demands of most current modular robotics approaches have led to highly complex modules. Similarly, the uncertainty associated with manipulating very small elements is challenging for deterministic manipulation approaches. Stochastic assembly processes have the potential to reduce these demands by taking advantage of probabilistic processes for various tasks, such as motion planning (Tomita et al., 1999; Ayanian et al., 2008; Varshavskaya et al., 2008; Werfel and Nagpal, 2008) or locomotion (Griffith et al., 2005; White et al., 2005; Napp et al., 2006; Gilpin et al., 2008). In the latter case, systems take advantage of stochastic environmental motions for the transportation of robot modules to target locations and/or the removal of unwanted modules. This approach reduces the requirement of module mobility at the cost of increased target structure design and assembly planning requirements.

The challenge of assembly planning under uncertainty is relevant to fields beyond modular robotics to general structure reconfiguration (Nagpal et al., 2003; Klavins et al., 2006; Lerman et al., 2006; Werfel et al., 2006; Hjelle and Lipson, 2009). In these situations, a target structure or system is assembled from a set of components whose availability or timing is uncertain, either due to resource fluctuations, unreliability of assembly operations, or other stochastic and distributed control aspects. For example, this situation may occur when many robots need to coordinate the joint construction of a structure. Since it may not always be possible to maintain perfect information on the state or location of individual robots, a planning approach that can accommodate such uncertainties is desirable.

In this paper we present an automated approach that plans the stochastic assembly of a target structure without knowledge of the times or locations of component availability. Unlike previous approaches, our proposed algorithm plans on-line during assembly, and is thus able to adjust to changing conditions, and give an estimate of the assembly time

[1]Cornell University, NY, USA

Corresponding author:
Michael T Tolley, Cornell University, 239 Upson Hall, Ithaca, NY 14853, USA.
Email: mtt33@cornell.edu

remaining. In addition, our algorithm accelerates stochastic assembly by finding multiple assembly paths that can be pursued in parallel. In order to accomplish this, our algorithm must, at each stage of the process, determine the next set of locations to allow assembly. This is achieved by sampling the graph of all possible paths from the current state to the target structure. For each of these samples, the assembly problem is solved by beginning with the final structure and working backwards, removing one accessible component at a time. As long as assembly constraints are taken into account during disassembly, each sample represents a valid path to the target structure. Thus, a path to error-free assembly is guaranteed at all times. For the case where the sampling finds more assembly paths than can be pursued in parallel (due to practical considerations), this approach affords the opportunity to choose the subset of paths that maximize assembly rates.

In the next section, we review previous work in stochastically reconfigurable systems that we believe may benefit from the assembly-planning approach presented here. We also discuss work in other fields that is related to our approach to planning for stochastic assembly. In Section 3, we specify the problem we aim to address, and our corresponding assumptions about the problem. We then present our on-line stochastic assembly-planning algorithm in Section 4, and describe its application to a particular type of stochastic assembly system in Section 5. Section 6 describes a set of simulations used to evaluate the performance of the assembly planner, along with the corresponding results. This is followed by a general discussion of these results and the strengths and weaknesses of the approach in Section 7, and overall conclusions drawn from the work in Section 8.

2. Previous and related work

Previous work has developed a variety of stochastically reconfigurable artificial systems that may benefit from the approach presented here. These systems can be grouped broadly into three categories: those that form assemblies from modules, with modules and assembled structures moving stochastically in an agitated environment (e.g. Figure 1(a)); those with modules in a stochastic environment depositing onto fixed assemblies (e.g. Figure 1(b)); and those with multiple decentralized robots manipulating structural elements (e.g. Figure 1(c)). In the first case, modules move about stochastically due to simulated Brownian motion and decide whether or not latch together when they collide into one another. This concept has been demonstrated experimentally in two dimensions by floating robotic modules on an air table, which is shaken to create stochastic motion (White et al., 2004; Griffith et al., 2005; Napp et al., 2006). These modules have electromagnets, actuated permanent magnets, or actuated latches on each face that are used to selectively bond modules together. The assembly and reconfiguration demonstrated by White et al. (2004) followed a hand-coded assembly plan. The modules of Griffith et al. (2005) ran hand-coded finite-state machines in order to reproduce linear strings of modules, or to produce repeating linear or planar patterns.

Napp et al. (2006) define graph grammars that are used to direct the assembly of their *Programmable Parts* into target structures. Various assembly and disassembly steps are treated like concurrent chemical reactions, the rates of which can be defined to result in the assembly of a desired structure. Napp et al. described an automated rule-synthesis procedure that, similar to the approach presented here, begins with a desired target shape and disassembles it one step at a time in order to determine a sequence of assembly steps. However, unlike in the current work, the assembly rules in Napp et al. (2006) are established off-line (prior to assembly). One of the challenges with this approach is the inability to compensate for changing conditions during assembly. For example, if the target shape or the various assembly rates change during assembly, the system has no way to adapt. In addition, while the approach in general allows multiple 'reactions' (assembly events) to occur simultaneously, the presented rule-synthesis procedure results in a single assembly sequence that allows assembly to occur at a maximum of two locations at a time. For a stochastic assembly system, this could severely limit the assembly rate of large structures that must wait for one of two very specific assembly events to occur.

A second type of stochastically reconfigurable system uses stochastic fluid agitation to transport modules to where they are selectively assembled to a fixed structure (White et al., 2005; Zykov and Lipson, 2007; Krishnan et al., 2009; Tolley et al., 2010). In this case, selective bonding between modules is achieved using permanent or electro-magnets, latches, and/or hydrodynamic forces. These efforts have demonstrated the assembly of a small number of modules on length scales spanning 500 μm to 13 cm. We have previously proposed a number of simple assembly-planning strategies for stochastic fluidic assembly (SFA), which we tested in simulation (Tolley et al., 2010). However, these approaches had difficulty achieving perfect assemblies without severely slowing assembly rates. A detailed description of this SFA approach is given in Section 5, as it will be the example application for our assembly-planning algorithm.

The third type of stochastically reconfigurable system involves passive structural modules that are deposited by active modules (Everist et al., 2004; Jones and Mataric, 2004; Terada and Murata, 2004; Detweiler et al., 2007; Hjelle and Lipson, 2009, Figure 1(c)). Werfel and Nagpal (2008) describe an assembly approach for such a system in which cubic structural modules are assembled by active modules. Their approach is to have the structural modules determine acceptable adjacent locations for new modules, which are transported by mobile robots. This is done by defining a set of rules for selecting block deposition locations that avoids leaving inaccessible gaps in the final

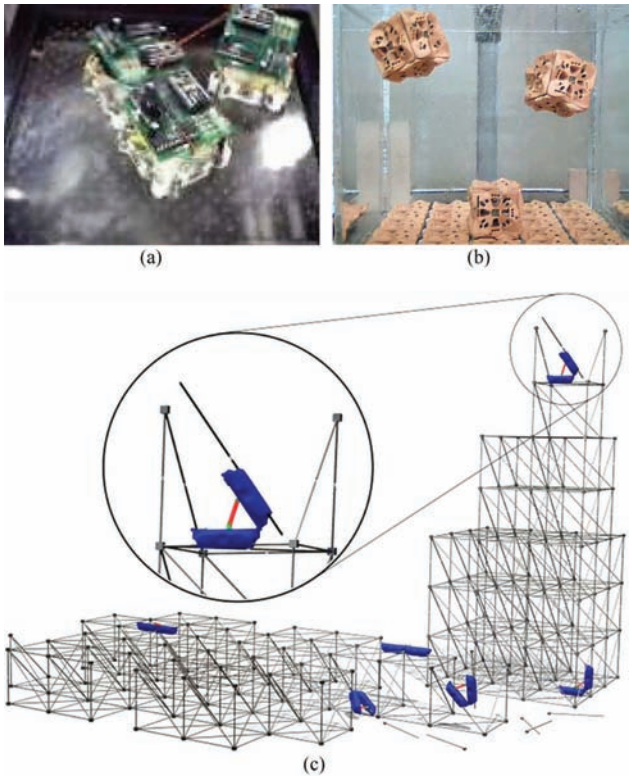


Fig. 1. Stochastic robotic assembly systems. Various types of stochastic assembly approaches have been proposed for robotic assembly. In previous work, we have investigated systems that assemble stochastically (a) on an air table with agitation (White et al., 2004), (b) in a fluidic tank (Zykov and Lipson, 2007), and (c) from structural elements manipulated by many simple truss-climbing robots (Lobo et al., 2009). Taking inspiration from nature, these approaches rely on stochastic processes for component transport and/or control. The present work investigates an assembly algorithm that can compensate for the unpredictable availability of components inherent in these systems.

structure. The robots then find locations to deposit their blocks by communicating with the structural modules as they climb over the structure. In order to minimize communication and make the robots as simple as possible, one control scheme they propose is to have the robots move about randomly until a suitable assembly location is encountered. However, even deterministic robot control schemes would most likely lead to unpredictable module arrival times and locations due to their decentralized nature.

In fact, even centrally controlled systems that are able to deterministically coordinate assembly with a small number of robots will most likely be forced to revert to stochastic decentralized control when they scale up to large numbers of interacting robots. Thus, we see that there are a wide range of proposed and existing robotic systems that require assembly planners capable of coping with stochastic component availability.

Aside from reconfigurable systems composed of active and passive modules, there is a variety of related previous work in self-reconfigurable robotic systems consisting only of active modules (Rus and Vona, 2001; Murata and Kurokawa, 2007; Yim et al., 2007). However, since the motion of the modules in these systems can be prescribed deterministically, most assembly-planning work has focused on the problem of coordinating the motions of the modules to follow a single assembly plan. Unlike stochastic systems, it is not necessary for deterministic systems to be able to pursue multiple assembly paths in parallel. Nonetheless, some authors have investigated assembly approaches with characteristics similar to stochastic assembly. For example, Tomita et al. (1999) describe a system in which free modules circulate continuously about assembled ones until they are identified to be in an appropriate location for the target structure. Similar to the approach of Napp et al. (2006) discussed above, each of the modules contains a pre-specified (static) assembly plan to achieve the target structure. In addition, the authors do not describe a general procedure for generating the assembly plan.

Fitch et al. (2003) describe a reconfiguration planner for a heterogeneous self-reconfiguring robotic system that must order the modules during reconfiguration such that each has a path to its location in the final configuration. Similar to the current work, the authors use the approach of starting with the final shape, and removing accessible modules, in this case to determine the ordering of the heterogeneous modules.

The approach of achieving modular robotic assembly by disassembly has also been taken more literally by Gilpin et al. (2008), who have developed a system that physically disassembles modules from an assembled mass to reveal the target structure. Since this approach is purely subtractive, it avoids the problem of assembly planning altogether. However, it is accompanied by other challenges, such as assembly of the initial mass, reconfiguration, and the assembly of hollow structures.

Aside from the disassembly approach, our planning algorithm shares properties with other previous work in path planning. For example, our approach of random sampling to identify the possible next steps in the assembly plan is similar to the central idea of Rapidly Exploring Random Trees (LaValle and Kuffner, 1999). Similarly, the overall approach is related to the concept of partial-order planning (Sacerdoti, 1975), in that the goal is to find many valid assembly sequences that can be pursued in parallel as opposed to following a linear plan.

3. Problem statement and assumptions

Our goal is to develop an on-line, high-level algorithm to plan the assembly of stochastically reconfigurable systems such as those described in the previous section. The relevant

common aspect of the systems discussed is the unknown temporal and spatial availability of assembly components. The goal is to be able to input a target configuration (i.e. shape map) and have the planning algorithm autonomously identify a path to error-free assembly. Due to unpredictable component availability, this algorithm must compensate for potentially slow assembly rates by pursuing multiple assembly paths in parallel. Thus, the algorithm must be able to identify a set of valid locations to which modules can be attached at any given stage of assembly. For the case where the number of possible concurrent assembly locations is limited by practical constraints, we also aim to develop heuristics for choosing the subset of possible assembly locations that maximize the overall assembly rate. Finally, we seek an on-line approach that can adapt to changes, such as a change in the target shape (assuming such a change is possible given the current configuration) or fluctuations in the assembly rates, and predict the assembly time remaining.

We assume kinematic constraints on assembly based on the local geometry of the assembly site (e.g. situations where a module could be assembled but does not have access). We also assume that there is some cost associated with every assembly event that varies from site to site (e.g. assembly takes longer based on the local geometry).

While we believe this algorithm could be useful for systems with various component morphologies – such as the truss assembly robots seen in Figure 1(c) (Hjelle and Lipson, 2009) – we limit the discussion here to cubic components that assemble on a rectilinear lattice due to the popularity of this morphology in modular robotics. Thus, the main kinematic constraint is the inability of the system to insert a component directly between two opposing components (Figure 2(a)). In the case of hollow objects, it is also necessary to verify that a free path exists between the module in question and the location of the module source (e.g. a boundary of the assembly volume). However, we ignore this case for the purposes of the present work. In addition, while we aim to develop an algorithm with applications beyond modular robotics, we will use the terms *module* or *cube* interchangeably to refer to the components of the system being assembled stochastically.

Assembly occurs on a plane, which we call the *ground plane*, to which initial modules attach. All modules must be connected to this ground plane either directly or indirectly through other modules at all times during assembly. Adjacent modules are assumed to be able to bond together when assembled and the loads on these bonds are not considered during assembly. This assumption may be valid in weightless environments (such as marine or space applications). For terrestrial applications, it may be possible (in future work) to include mechanical loading in the assembly cost calculation. Finally, we do not consider the possibility of removing modules during assembly (e.g. for scaffolding).

4. On-line stochastic assembly-planning algorithm

4.1. Assembly sequence generation

One approach to achieve the goals described in the previous section is to establish a set of conditions that can be checked to verify if the addition of a module at a given location will lead to an unfillable gap in the structure (Werfel and Nagpal, 2008). However, there are two drawbacks with this approach. Firstly, it is difficult to enumerate a simple set of conditions that can be proven to guarantee the assembly of a target structure without placing restrictive limitations on its shape (such as having no loops or holes in any plane). Secondly, for the case where only a limited number of assembly sites can be activated in parallel, this approach gives no indication of how desirable any particular valid location is with respect to any others. While a certain location may be valid in that it does not prevent successful assembly, it may constrain further assembly in an undesirable way.

In order to address these drawbacks we look at the problem from another direction. This approach is inspired by the experience of how much easier it is to take a puzzle apart than to put it back together. Instead of solving the hard problem of determining where to *add* modules without leading to difficulties further down the line, we start with the completed structure and solve the easier problem of finding modules to *remove*. As long as the constraints on addition, such as not inserting a cube directly between two others, are applied in reverse during removal (cannot remove a cube from directly between two others), simply reversing the sequence of removals gives a valid sequence of additions (see Extension 1). This circumvents the problem of determining what complications a particular addition will cause in the future, because we already know what the future looks like. We can get an estimate of the expected assembly time for this sequence by adding up the expected time for the reverse of each removal step (calculated based on the local geometry).

This process for obtaining an assembly sequence by disassembling a target structure is described in Algorithm 1. This algorithm takes a representation of a target structure to be assembled and returns a valid assembly sequence and the associated expected assembly time. For each iteration of the main *while* loop, the algorithm selects random modules in the structure until it finds one that is both accessible and does not disconnect the structure when removed. A module is accessible if it could have been assembled last subject to assembly constraints.

The disconnection check is an important part of this algorithm. Essentially we are checking if the module in question is an articulation point of the connectivity graph. Without it, structures could violate the requirement of being connected at all times during assembly. Similar checks are necessary in non-stochastic modular robotic reconfiguration

Algorithm 1: Assembly sequence generator.

```
initialize structure with user shape map
while modules remain in structure do
  candidate-module ← random module in structure
if candidate-module is accessible then
  if NOT RemovingModuleDisconnectsStructure
  (candidate-module, structure) then
    add candidate-module assembly cost to cost-sum
    add candidate-module location to sequence-list
    remove candidate-module from structure
  end if
 end if
end while
  reverse sequence-list to obtain assembly sequence
  RemovingModuleDisconnectsStructure(module,
  structure)
  neighbors ← list of modules adjacent to module in
  structure
  if SizeOf(neighbors) = 1 then
    return false
  end if
  neighbor ← random neighbor from neighbors
  neighbors ← (neighbors – neighbor)
  structure ← (structure – module)
  num-connected-neighbors ← 1
  breadth-first search (BFS) through each
  connected-module in structure connected (directly or
  indirectly) to neighbor:
    if connected-module{QUOTE ∈}neighbors then
      increment num-connected-neighbors
      if num-connected-neighbors =
      SizeOf(neighbors) then
        return false
      end if
    end if
  end BFS
  return true
```

(Rus and Vona, 2001). Our approach to checking connectivity is to remove the module in question and check if its neighbors remain connected to one another through other modules and/or the ground plane. This problem is simplified by the following two observations.

If there is only one neighbor, then there are no parts to be disconnected, thus no check is necessary.

If there are more than three neighbors, then this module is not accessible to begin with. This is because it is not possible to choose four cubes neighboring a given cube without two of the chosen cubes being opposite one another (e.g. Figure 2(b)), in which case the cube is inaccessible.

Thus, we need only consider the two- and three-neighbor cases. Our approach is to start with a random neighbor and perform a breadth-first search through connected modules to find the other neighbor(s). For the purposes of this search, we consider the ground plane as a module with many possible connections. This search ends when either all neighbors are found to be connected (thus no disconnection), or all connected modules have been searched and all neighbors were not found (thus disconnection). A breadth-first search is used since in a majority of cases, the neighbors will be connected within a few steps by nearby cubes. The worst-case scenario is the removal of the only module connecting the ends of long protrusions (e.g. Figure 2(c)). However, in general we expect the disconnection check to take a small number of steps.

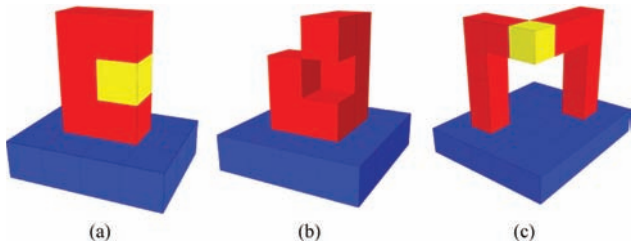


Fig. 2. Example configurations. (a) The highlighted module could not be added last since it is impossible to insert a module directly between two others (above and below, in this case). Thus, the highlighted module could not be removed first by Algorithm 1. (b) It is impossible for a module to be inserted into a location with four existing adjacent modules. (c) The worst-case situation for a breadth-first disconnection check function is a module – such as the highlighted one seen here – that connects two long protrusions.

In order to prove the effectiveness of Algorithm 1, we require the following four definitions and two lemmas.

Definition 1. An *admissible structure* is a finite-sized, connected structure composed of regular cubes assembled on a rectilinear lattice.

Definition 2. A *boundary* is an object that occupies the space of one or more connected cubes in a single layer, and is connected to an admissible structure on exactly one plane, but is not considered part of the admissible structure.

Note that the ground plane upon which assembly occurs is considered a boundary.

Definition 3. A structure's cube is *accessible* if it could have been assembled last subject to assembly constraints.

This definition, along with our assumption that the only constraint on assembly is that a cube cannot be inserted directly between two others, or between a cube and a boundary, leads to the following corollary.

Corollary 1. A cube without neighbors or a boundary on at least three adjacent sides is accessible.

Definition 4. The cube of a structure connected to a ground plane is *removable* if it is both accessible and can be

detached without separating the remaining structure into pieces that are disconnected from both each other and from the ground plane.

Lemma 1. An admissible structure connected to a boundary will have at least one accessible cube.

Proof. We prove Lemma 1 by construction. For the purposes of this proof we first define the coordinates x, y, and z such that the structure connects to the boundary at the $z = 0$ plane with the boundary occupying the negative-z side (see Figure 3(a)). Since admissible structures are finite sized, there exists some plane of cubes perpendicular to the z-axis that contains at least one cube in the structure, but beyond which there are no more cubes in the positive z-direction (see Figure 3(b)). Likewise, there exists a row perpendicular to the y-axis in this plane that contains at least one structure cube, but beyond which there are no more cubes in the positive y-direction (see Figure 3(c)). Finally, in this row there exists some cube beyond which there are no more cubes in the positive x-direction (see Figure 3(d)). This cube has no neighbors in the positive x, y, or z directions, and is thus accessible by Corollary 1. □

Lemma 2. If an admissible structure can be separated into disconnected pieces by the removal of an accessible cube (i.e. the cube was an articulation point of the connectivity graph), these pieces, considered individually, are themselves admissible structures.

Proof. Lemma 2 can be easily proven by contradiction since any piece of a finite-sized, connected structure composed of regular cubes assembled on a rectilinear lattice must also share these properties. □

Lemma 3. Any admissible structure connected to a ground plane will have at least one removable cube.

Proof. We prove Lemma 3 by construction. By Lemma 1, the given admissible structure must have at least one accessible cube (see Figure 3(e)). Removing this cube will either separate the structure into one or more pieces disconnected from each other and the ground plane (Case 1), or it will not (Case 2). For Case 2, the lemma has been proven. For Case 1, we consider the substructures that would have been created if this cube were removed. At least one of these substructures is not connected directly to the boundary and is connected to the rest of the original structure only at the chosen cube. In addition, this substructure contains at least one cube less than the original structure. By Lemma 2, this substructure is itself an admissible structure. If we consider the chosen cube to be this substructure's boundary, we can now, by Lemma 1, find an accessible cube in this substructure (see Figure 3(f)). We are now left with either Case 1 or Case 2 as described above. If Case 1 is encountered, we simply repeat this process, each time reducing the size of the substructure under consideration by at least one cube (Figure 3(g) and (h)). Eventually, if a Case 2 is not encountered, the substructure under consideration will consist of only one cube, which is removable (Figure 3(i)). □

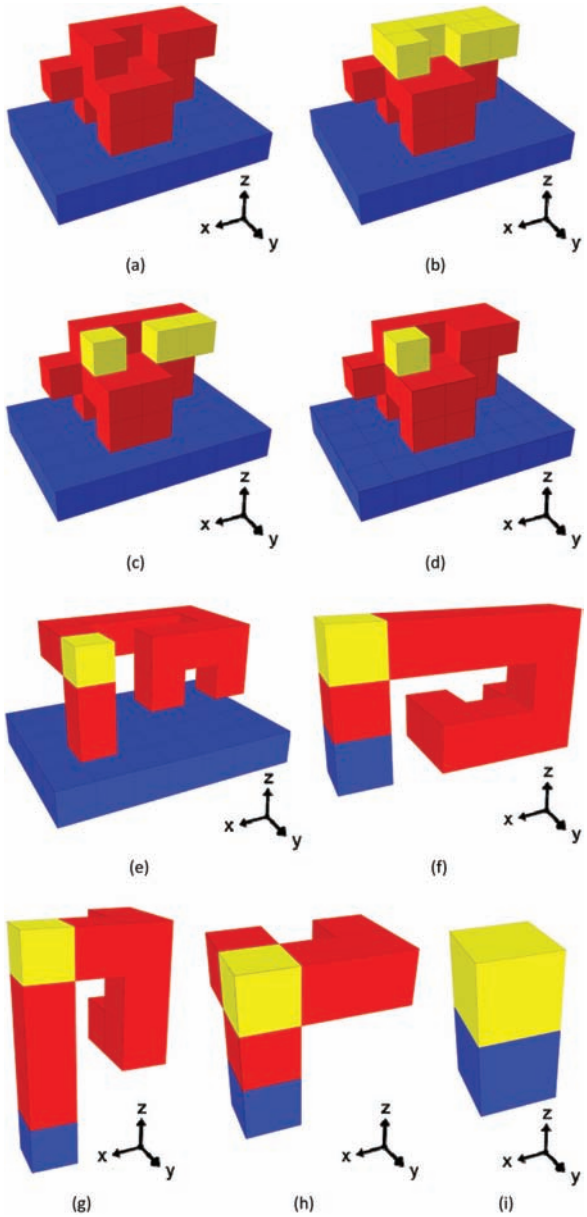


Fig. 3. Finding an accessible cube that does not disconnect the target structure. (a) Given a target structure (red cubes) attached to a boundary (blue cubes), an accessible cube can be found by (b) considering the top layer of the structure, (c) moving to the front row, then (d) selecting the leftmost cube in this row. (e) If removing an accessible cube found this way (yellow) would disconnect the structure, (f) perform the same procedure on the substructure consisting of one of the substructures that would be disconnected. (g)-(i) Following this procedure recursively will, in the worst case, eventually find a single-cube substructure that is, by definition, accessible (color online only).

Theorem 1. Algorithm 1 will find a finite, error-free assembly sequence for any given admissible structure attached to a ground plane.

Proof. We prove Theorem 1 by induction. By Lemma 3, at the first stage of disassembly, there exists at least one removable cube. Thus, choosing and testing structure cubes randomly without replacement, Algorithm 1 will find a removable cube at the first stage of disassembly. With the first cube removed, the structure is now either empty (Case 1), a connected structure attached to the ground plane (Case 2), or a set of connected structures each attached to the ground plane (Case 3). In Case 1 we have completed a disassembly sequence. In Case 2, the remaining structure is an admissible structure attached to a ground plane with one less cube than the original. In Case 3, by Lemma 2, each of the remaining structures is an admissible structure attached to the ground plane, and the sum of their constituent cubes is one less than that of the original structure.

By Lemma 3, the remaining structure(s) in Case 2 and Case 3 each has at least one removable cube. The same is true for all further substructures revealed by the removal of these cubes. Since the original structure is finite sized, there are a finite number of these steps required to disassemble the complete structure. The sequence composed of the reverse of each disassembly step in reverse order is a finite, valid assembly sequence that contains every module in the given structure. □

4.2. Sample-based parallel assembly

Algorithm 1 is guaranteed to find one possible sequence for the assembly of a given target structure. However, this alone does not solve the problem of planning for stochastic assembly because waiting for a module to arrive at one particular location could take a prohibitively long time (particularly if the transportation mechanism is random motion). In order to accelerate this process we would like to be able to pursue many potential assembly sequences in parallel.

To achieve parallel assembly, we follow a sampling approach that iterates Algorithm 1 many times to map out potential assembly paths. During sampling, we keep track of the location of the last module removed in each sequence, as well as the sequence's expected assembly time. This sampling gives a set of possible locations to next add modules. We then select from among these the set of locations to allow or promote assembly, which we call *assembly promotions sites*. Ideally, we would want to promote assembly at all of the locations found during sampling. However, due to practical constraints on the system, there may be a limit on the total number of simultaneous assembly promotion sites. Nonetheless, pursuing assembly at multiple sites in parallel significantly reduces the expected assembly time, because the overall parallel assembly rate is the sum of each of the serial assembly rates (assuming they are independent).

In essence, this procedure is sampling the graph of all possible assembly sequences from the current state to the goal state (Figure 4). While this graph could theoretically be enumerated in its entirety, in practice it is prohibitively expensive to compute for all but the simplest structures.

Algorithm 2: Parallel stochastic assembly planner.

At the start of assembly:
initialize *target* with user shape map

During each assembly stage:
update *assembled* based on current configuration
remaining ← (*target* – *assembled*) (the relative compliment)
for S (number of samples) **do**
  call Algorithm 1 on *remaining*
  increment *count* of final location in sample's *sequence-list*
  add sample's *cost-sum* to final location's *total-cost*
end for
select *N* locations to promote assembly

Stochastic process:
selects location of next module arrival
assembled is updated

The price to be paid for the increased assembly rate of this parallel approach is a decrease in the certainty about the assembly sequence. In addition to not knowing the time of the next assembly event, we now also do not know its location. Thus, the stochastic process 'chooses' which of the selected promotion sites will be filled at each stage of assembly and our algorithm must be able to cope with this additional uncertainty. This is achieved by performing the same sampling for each stage of assembly, in order to choose the next set of promotion sites. This is also the feature that allows our planner to adjust to changes that occur between assembly events. Algorithm 2 details this approach, which is illustrated in animation in Extension 1.

The process begins with the user specifying a target shape map. Assembly then proceeds by alternating between Algorithm 2 selecting a set of assembly promotion sites, and the stochastic process selecting from among these sites, the location of the next assembly event. Figure 5 graphically depicts the entire parallel stochastic assembly process that employs this algorithm.

In stages beyond the first, when Algorithm 1 is called by Algorithm 2 calls for sampling, it selects from among only the remaining cubes for removal, but considers the previously assembled cubes for accessibility and disconnection checks. Since Theorem 1 requires an admissible structure to be assembled on a flat ground plane, we present Theorem 2 to addresses the correctness of Algorithm 1 in this case.

Theorem 2. Algorithm 1 will find a finite, error-free assembly sequence for the remaining part of an admissible structure attached to a ground plane that has been partially assembled by Algorithm 2.

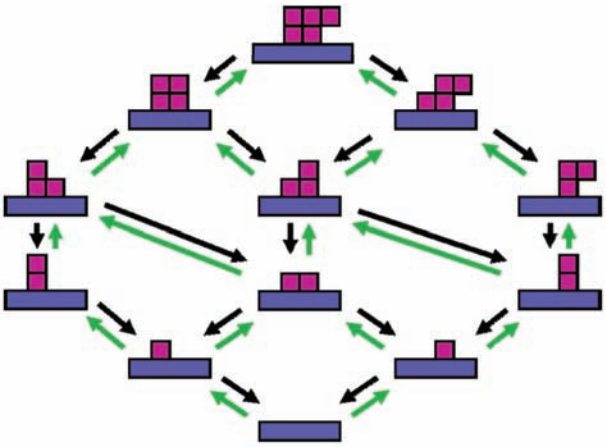


Fig. 4. Assembly by disassembly. The parallel stochastic assembly planner samples the graph of all possible paths to assembly by starting with the complete structure and removing one cube at a time until the existing state is revealed (dark arrows). Reversing the disassembly sequence reveals a valid assembly sequence (light arrows). Assembly promotion locations are chosen based on these samples.

Proof. We prove Theorem 2 by induction. The base case is the first assembly stage where the set of previously assembled cubes is empty. In this case, Algorithm 1 will find a finite, error-free assembly sequence by Theorem 1. Now if we assume for an assembly stage N that at least one finite, error-free assembly sequence was found, Algorithm 2 follows the first step of one such sequence to reach stage $N + 1$. The remainder of the sequence found in stage N (i.e. without the first step), is a finite, error-free assembly sequence for the structure remaining in stage $N + 1$. Algorithm 1, choosing cubes without replacement and checking them for accessibility and removability (taking the previously assembled cubes into account), is guaranteed to find such a sequence in stage $N + 1$. By induction, Theorem 2 holds for all assembly stages. □

Algorithm 2 has many desirable properties that make it suitable for stochastic assembly. Firstly, it is an on-line approach, as it re-evaluates progress at each stage of assembly. This allows the algorithm to pursue many potential assembly paths in parallel, which greatly increases assembly rates. In addition, this has the potential to allow the approach to compensate for the unforeseen circumstances that may occur during stochastic assembly, such as changes in the assembly rates or the target shape. As long as the inputs to Algorithm 2 are updated, it will find assembly paths based on the most recent information.

A second desirable property of Algorithm 2 is that in addition to its ability to find parallel assembly sequences, it gives us the opportunity to maximize assembly rates based on the selection of locations to allow assembly. It also gives us the opportunity to collect statistics during the sampling of the remaining structure in order to estimate the distribution of assembly rates and use this distribution to guide the maximization. We discuss potential promotion site selection heuristics in the following section.

Another advantage of the stochastic assembly-planning algorithm is its potential for adaptation to different types of stochastic assembly systems, such as those discussed in Section 2. In Section 5, we discuss the application of this algorithm to a SFA system. However, by tuning various parameters, such as the expected assembly rates and associated local structure configurations, as well as the number N of simultaneous attractions, we believe this algorithm could be applicable to a wide range of stochastic assembly situations. We investigate the effects of changes in some of these key parameters in Section 6.

Finally, this stochastic planning algorithm is able to adjust the amount of computational effort exerted during planning to adapt to the available time and resources. We expect increased sampling of the potential assembly sequences to give us a more accurate estimate of the relative merits of potential assembly paths (at the cost of increased computational effort). Our sampling approach provides a straightforward method of balancing between these tradeoffs. We examine such tradeoffs in simulation in Section 6.

4.3. *Assembly promotion site selection*

We investigated two possible assembly promotion site selection heuristics. We used these heuristics to select the attraction locations from among the set of possible locations found during sampling, based on data collected during sampling. The *quickest paths heuristic* selects the potential modules with the quickest expected assembly times. The expected assembly times for each potential location are found by averaging the expected assembly times of all of the samples that removed the module in question last.

The second promotion site selection heuristic is the *most paths heuristic*. This heuristic is based on the conjecture that our random sampling process will tend to favor locations that maximize the number of paths to assembly. In other words, promotion sites that restrict the number of ways the remaining structure can be assembled will tend to be sampled less than those that lead to many possible assembly paths. We examine the relative effectiveness of these heuristics in Section 6.

5. Example application: stochastic fluidic assembly

In this section we describe an example stochastic assembly system in order to give a concrete description of how our algorithm could be applied to achieve on-line assembly planning. Our example system uses stochastic fluid motion to enable the assembly of minimalistic robotic modules. We have investigated this approach with physical and simulated systems of various scales (Figure 6). In this system, a target

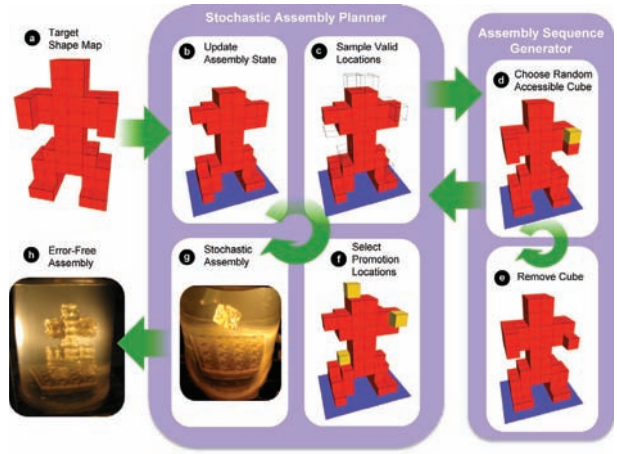


Fig. 5. Diagram of parallel stochastic assembly process. (a) User inputs the target shape map. (b) The stochastic assembly planner updates the current state of assembly. (c) The assembly planner calls the assembly sequence generation algorithm to sample assembly sequences for the remaining structure. (d) The assembly sequence generator determines a valid assembly sequence by alternately choosing and (e) removing random, accessible cubes that do not disconnect the structure when removed. (f) Once the sequence generator returns a set of valid assembly locations, the planner chooses from among these a set of promotion sites using some selection heuristic. (g) The stochastic process then determines which of the promotion sites next receives a module. (h) If there remain modules to be assembled, the process returns to step (b), otherwise, assembly is complete.

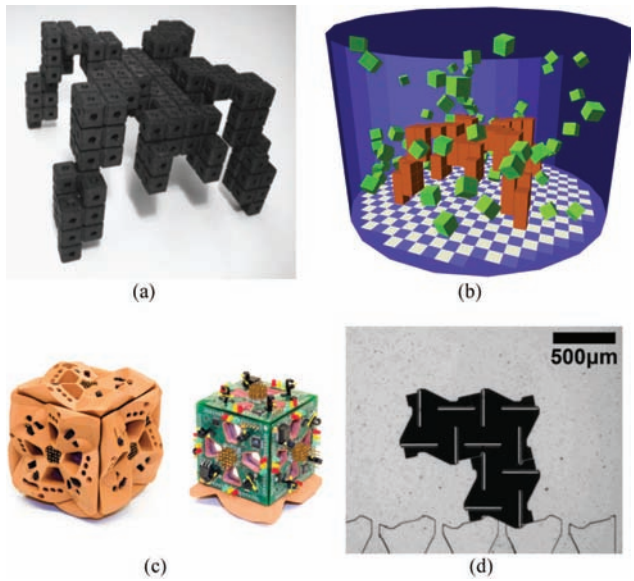


Fig. 6. Stochastic fluidic assembly. Modules are assembled by taking advantage of their stochastic motions within a fluidic system for transportation. A target shape (a) is assembled in simulation (b), where free and attached cubes are colored dark and light, respectively (Tolley et al., 2010). The assembly substrate is displayed as a checkered floor. (c) Centimeter (Zykov and Lipson, 2007), and (d) micrometer-scaled (Tolley et al., 2008) versions of stochastic fluidic assembly modules.

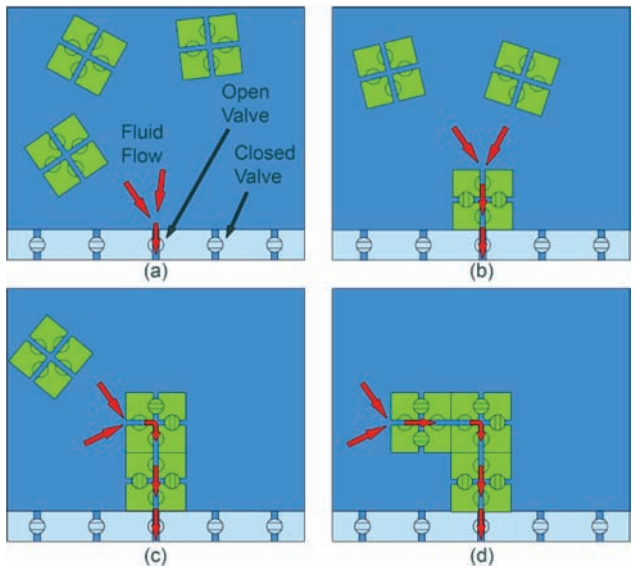


Fig. 7. Stochastic Fluidic Assembly Concept (Tolley et al., 2010). (a) Fluid flow (indicated by arrows) into a substrate attracts a nearby module moving stochastically in fluid environment. (b) Once attached, the module draws power from the substrate to activate on-board valves and redirect fluid flow through internal channels, (c) attracting new modules at desired locations. (d) This process continues until the structure is complete.

structure is 'grown' by adding modules first to a planar substrate, then to the exposed surfaces of previously assembled modules (Figure 7).

The modules are homogenous, and their availability for assembly at any particular location at any point in time is determined by the stochastic motion of the fluid in the assembly chamber. An assembly promotion site can be activated by redirecting fluid flow through the internal structure to open a sink at any surface of the structure, which pulls in nearby modules until one comes close enough to attach. However, there is a limit to the number of sinks that can be opened simultaneously (due to the increased flow rates required to maintain additional sinks). In addition, the attraction probability is affected by the local fluid flow, which in turn is affected by the local assembled structure. In fact, the fluid flow is highly coupled with the existing assembled cube configuration, but we assume here for simplicity that the locations immediately adjacent to a sink have the largest impact. Thus, the expected cube arrival time is a function of local geometry only.

Due to kinematic constraints, there are three possible configurations to which a cube can be attracted (Figure 8(b)). In order to determine the probability distributions for cubes being attracted to these three basic configurations over time, we conducted a set of experiments with a test fluidic system (Figure 8). Each trial was conducted starting with four 1 cm cubes moving stochastically in water

in a 15 cm tall by 6 cm radius cylindrical tank. The fluid agitation was achieved by pumping water out through two inlets at the top of the tank, and back in through a jet near the middle of the tank at approximately 1.5 l/min (as indicated in Figure 8(a)). At a random point in time, a valve was opened to allow fluid to flow out through the sink(s) at the target location. We then measured the time until one of the free-floating cubes was attracted to the target location. This experiment was repeated 50 times for each of the three configurations.

The results of these experiments are shown in Figure 8(c), which shows histograms of the durations of the trials, binned into one second intervals up to 35 s. We see that the assembly times were roughly exponentially distributed, with fast assembly times being more likely, but much longer assembly times also being possible. Overlaid on the experimental results are the most probably exponential fits. Figure 8(d) compares the mean assembly times for the three cases. It is important to note that these experiments investigate only the time required to attract a module to the right location and do not take into account any additional time required to engage latching mechanisms and establish communication between modules. Nonetheless, these results support our assumption that the assembly rates in our system depend on the local structure configuration. In addition, they provide us with a lower bound on assembly time distributions for testing our assembly algorithm. Using these three expected assembly times, we can create a simplified model of our system using Gillespie's stochastic sampling algorithm (Gillespie, 1977; Gibson and Bruck, 2000).

6. Simulations and results

We employed the method described in the previous section to simulate our stochastic fluid assembly system. This simulation proceeds by alternating between two steps, as described in Algorithm 2. After initialization of the assembly planner with the desired user shape map, in the first step of the simulation, the planner finds a set of allowable promotion sites and selects a subset of these to promote assembly. The second step of the simulation uses Gillespie's method to randomly select the time and location of the next assembly event, according to the distributions determined by the expected assembly times of the locations in question. These expected assembly times, in turn, are determined by the local structure configuration. The simulation proceeds by alternating between these two steps of assembly planning and random selection until the structure is complete (see Extension 1). In the rest of this section we describe results of our investigations into the effects of changing the location selection heuristic, as well as the number of parallel promotion sites and path samples for each assembly stage. Finally, we analyze the scaling of the computational effort exerted by our algorithm with the number of modules in the target structure.

6.1. Effect of promotion site selection heuristics

We ran the simulation described above on the assembly of a hand target shape composed of 619 modules (Figure 9(a)). We set the number of simultaneous attractions to four and sampled the remaining structure 40 times at each assembly stage. We used the experimentally determined expected assembly times for the three assembly cases and calculated the total assembly time for each run. Figure 9(b) shows a comparison of the mean assembly time per cube (averaged over 150 runs), for the two promotion site selection heuristics described in Section 4, along with a random baseline (where assembly promotion locations are chosen at random from among the valid locations found by the assembly planner). We found no statistical decrease in the assembly times for the heuristics as compared to the random baseline.

One possible explanation for this null result in the effectiveness of the promotion site selection heuristics is that the experimental assembly times for the three location configurations (Figure 8(b)) are too similar to see a significant improvement, even for a large number of trials (150). However, it is possible to imagine a system with a large difference in the expected assembly times for various local configurations. For example, this might occur if a stochastic robot assembler has difficulties assembling a cube to a particular local geometry. As another example, a particular configuration may often lead to connection issues that require rejecting the module and waiting for a new one to arrive.

In order to evaluate the heuristics in such a case, we prescribed three sets of expected assembly times with one of the configurations seen in the Figure 8(b) set to be much more expensive than the other two. In the first situation, the expected assembly time for one outlet (configuration 1) was set to 100 s, while the other two configurations (2 and 3) had expected assembly times of 1 s. Likewise, we also investigated situations where either configuration 2 or 3 was 100 times more expensive. Figure 9(b) shows the results of these simulations. We see that there is now a significant difference in the average assembly times produced by the three heuristics. In all three cases, the *Quickest Paths* heuristic was found to lead to the fastest assembly times (or at least be tied with one of the other two heuristics for fastest assembly times). Conversely, the *Most Paths* heuristic actually increased assembly times in two cases with respect to random selection.

An examination of the ratios of the three assembly configurations encountered by the different heuristics showed that the *Quickest Paths* heuristic has the effect of reducing the number of assemblies to the expensive configuration, as compared to selecting the promotion sites randomly. By contrast, the *Most Paths* heuristic – which does not take assembly times into account – actually increases the frequency of the expensive configuration in Case 1 and Case 2. Thus, any benefits due to maximizing the number of promotion sites per assembly step are outweighed by the expensive

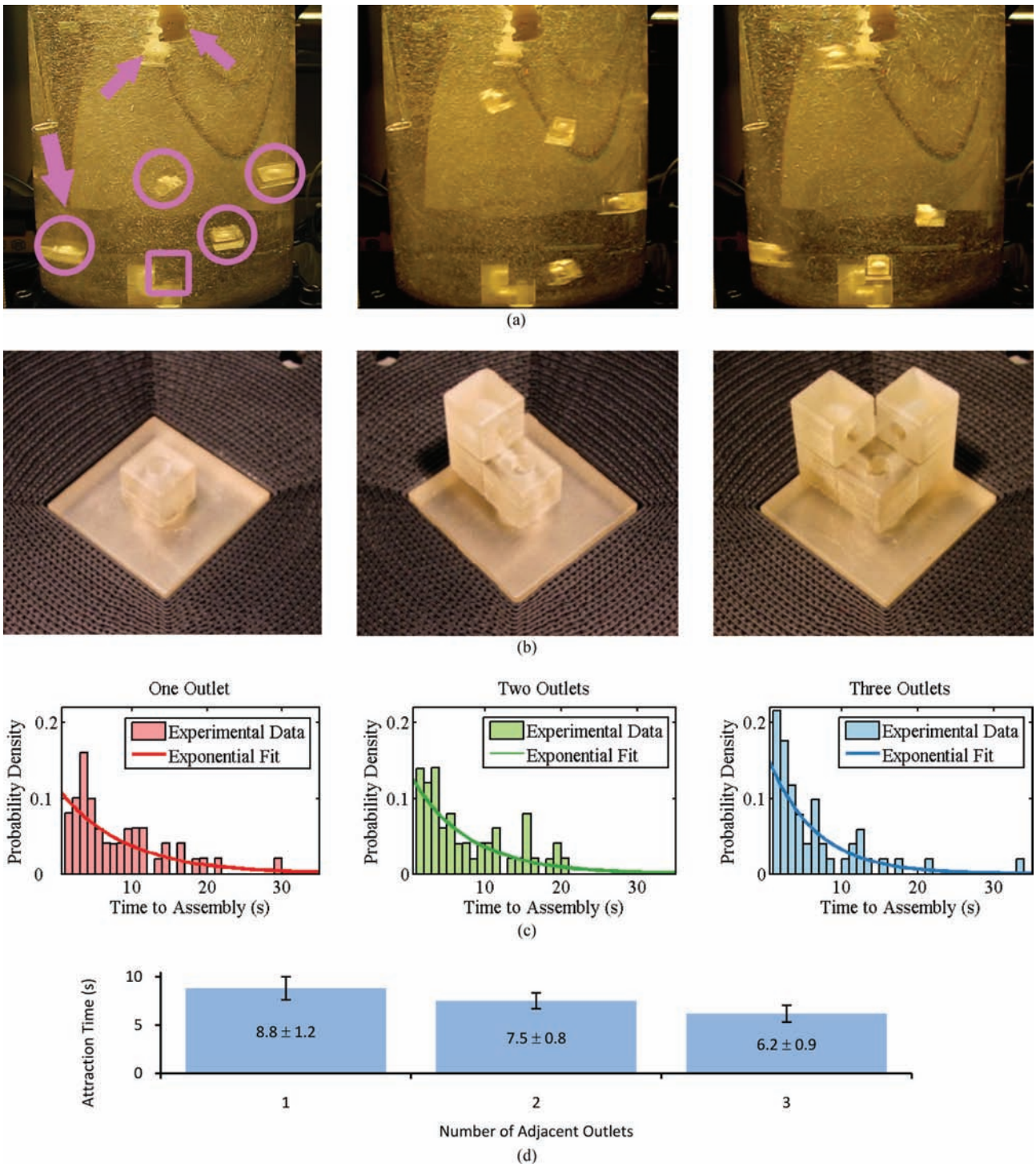


Fig. 8. Experimental determination of assembly probability function. (a) Experiments were conducted to record the time required to attract a free module undergoing stochastic agitation to one of three outlet configurations. The large arrow indicates inflow, smaller arrows indicate outflow, circles identify 1 cm cubes undergoing stochastic agitation, and the square denotes the target location to which cubes are attracted. (b) The three possible outlet configurations to which a new module can be attached. (c) Histograms of the time to attraction of 50 experiments for the three configurations, with exponential fit overlaid. (d) Comparison of the mean assembly times for the three cases, with the standard errors indicated.

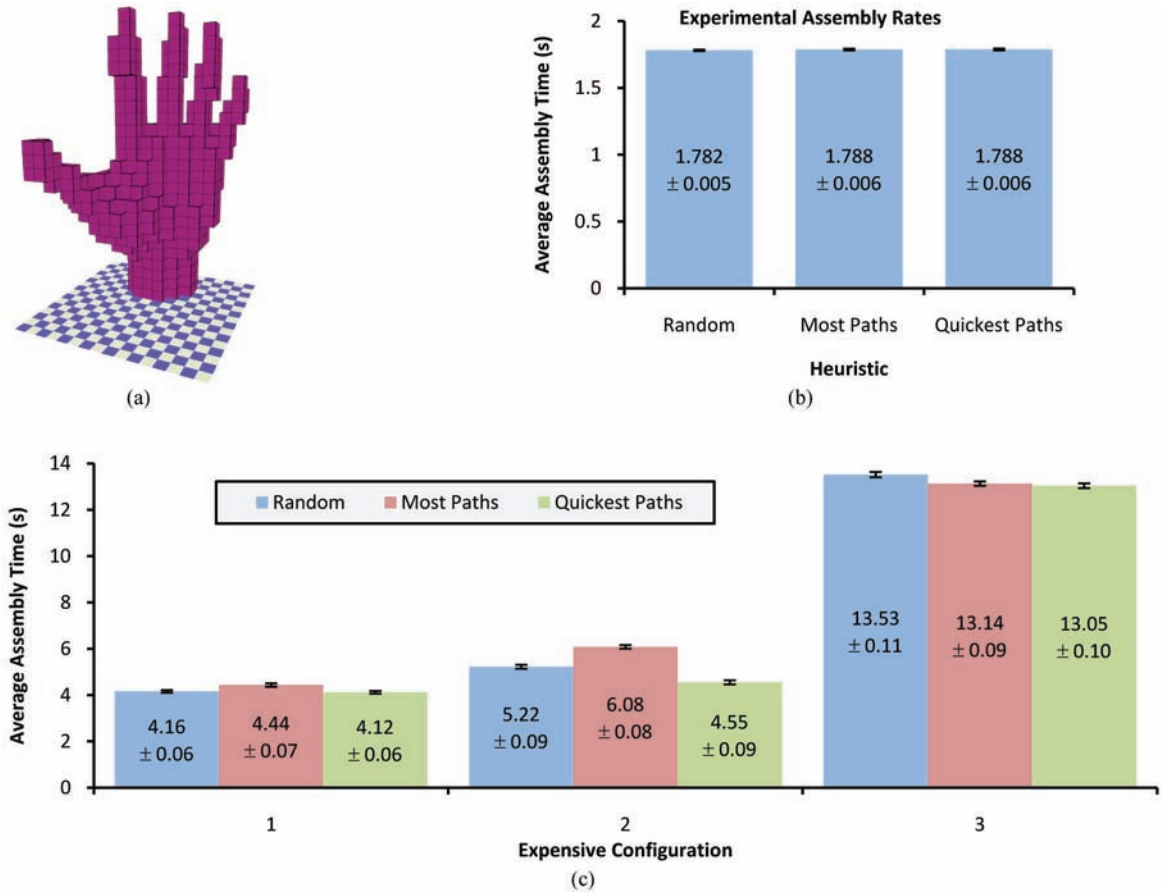


Fig. 9. Assembly promotion site selection. Two heuristics for choosing the subset of viable sites at which to encourage assembly were tested on the assembly of a 619-module hand-shaped target (a). (b) Simulations using experimental assembly rates found no improvement with these heuristics as compared to random site selection. (c) However, in situations where assembly to one of the three configurations seen in Figure 8(b) takes 100 times longer than the other two configurations, the *Quickest Paths* heuristic shows a significant improvement with respect to the *Most Paths* heuristic or random selection.

assembly rates of the assembly configurations preferred by this maximization.

6.2. *Effect of target shape*

In order to evaluate the dependence of the promotion site selection heuristic results on the target shape, we reproduced the results for one of the cases described in Section 6.1 on four additional, topologically different shapes of various sizes (Figure 10). We chose the expensive two-outlet configuration case, since it showed the most significant difference for the three heuristics (as seen in Figure 9(c)). For each of the new shapes, the average assembly time for the *Quickest Paths* heuristic was significantly shorter than for either the *Most Paths* or *Random* alternatives. In addition, in each case, the *Most Paths* heuristic's average assembly time was longer than random, except for the *Eiffel* shape, which was not significantly different. Thus, this result seems to be consistent across a range of target shapes and sizes.

6.3. *Effect of number of samples*

Since our assembly planner re-samples the remaining structure at each stage of assembly, the number of samples of the remaining structure performed at each stage of assembly obviously has a large impact on the computational effort exerted during assembly planning. Thus, we would like to know how many samples are necessary to minimize assembly times (at least to a local minimum). In order to investigate this, we ran simulations as described above, with four simultaneous promotion sites, and varied the number of samples. Figure 11(a) compares the mean assembly time per cube, averaged over 50 runs, for cases with 10, 20, 40, and 60 samples per assembly stage. The *Quickest Path* heuristic is compared against random promotion site selection for both the experimental assembly rates and the test rates with an expensive two-outlet configuration.

The first thing we see from this comparison is that, as in the results of Section 6.1, the *Quickest Paths* heuristic out-performed random location selection for the test values, while there was no reduction for the experimental values,

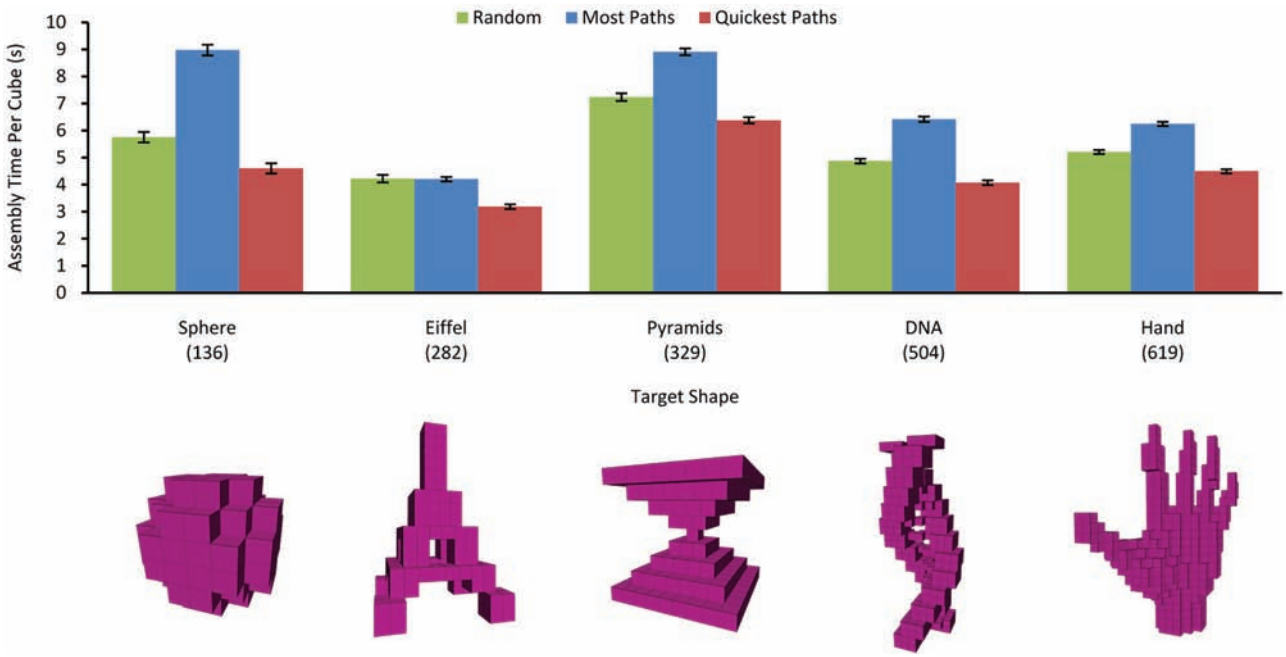


Fig. 10. Effect of target shape. The simulations performed to determine the effectiveness of the various promotion site selection heuristics on the assembly of the hand target shape (Figure 9) were repeated with four other targets of varying shapes and sizes. The numbers in parentheses indicate the number of modules in each shape. The relative effectiveness of each of the three promotion location selection heuristics was similar in each case.

regardless of the number of samples. In addition, for the test case, the average assembly time per cube decreased as the number of samples increased from 10 to 40, even with random location selection. This can be explained by the fact that 40 samples are more likely than 10 to find at least four different potential promotion sites at each assembly stage. Thus, for the 10 sample case, assembly may be forced to take an expensive assembly path when more sampling would have found a quick path to be pursued in parallel. As for a set of parallel resistors, a single low-resistance path significantly reduces the overall resistance. The experimental assembly rates, however, were similar enough that this effect did not significantly alter assembly times. In any case, this effect diminished as the number of samples increased from 40 to 60, indicating that there is an ideal number of samples (approximately 40 in this case), above which the additional computational effort does not result in reduced assembly times.

6.4. *Effect of number parallel promotion sites*

We were also interested in the effect of the number of parallel assembly promotion sites. While this parameter is most likely determined by practical system limitations, it may be useful to know how beneficial it would be to increase the number of parallel promotion sites. To this end, we conducted another set of simulations in which we varied the number of parallel promotion sites for both the experimental and expensive two-outlet test case scenarios (Figure 11(b)). For these simulations, we kept a constant ratio of 10 samples per promotion site, since this was found to be ideal in Section 6.3. As expected, the average assembly time per cube decreases as the allowed number of parallel sites increases. This was the case for both the experimental and the test cases, and it is, of course, the reason for pursuing parallel assembly to begin with. However, we see diminishing returns as the number of promotion sites increases. This is most likely due to the finite number of potential promotion sites at any given stage of assembly, as determined by the target shape (i.e. increasing from six to eight allowed promotion sites does not help in assembling a structure with a maximum of six valid promotion sites at any given assembly stage).

As before, we found no difference between the heuristics for the experimental case. However, there was an interesting decrease in the effectiveness of the *Quickest Paths* heuristic with respect to random site selection. Figure 11(c) shows the reduction in assembly time due to the *Quickest Paths* heuristic, as a percentage of the baseline assembly time. We see that as the number of promotion sites increases from two up to 10, this reduction decreases from 20% down to 5%. Thus it seems that the advantage of choosing the promotion sites carefully decreases as the fraction of the total possible sites chosen increases.

6.5. *Estimating assembly time*

We would like our on-line assembly algorithm to be able to estimate the amount of time required to assemble a given

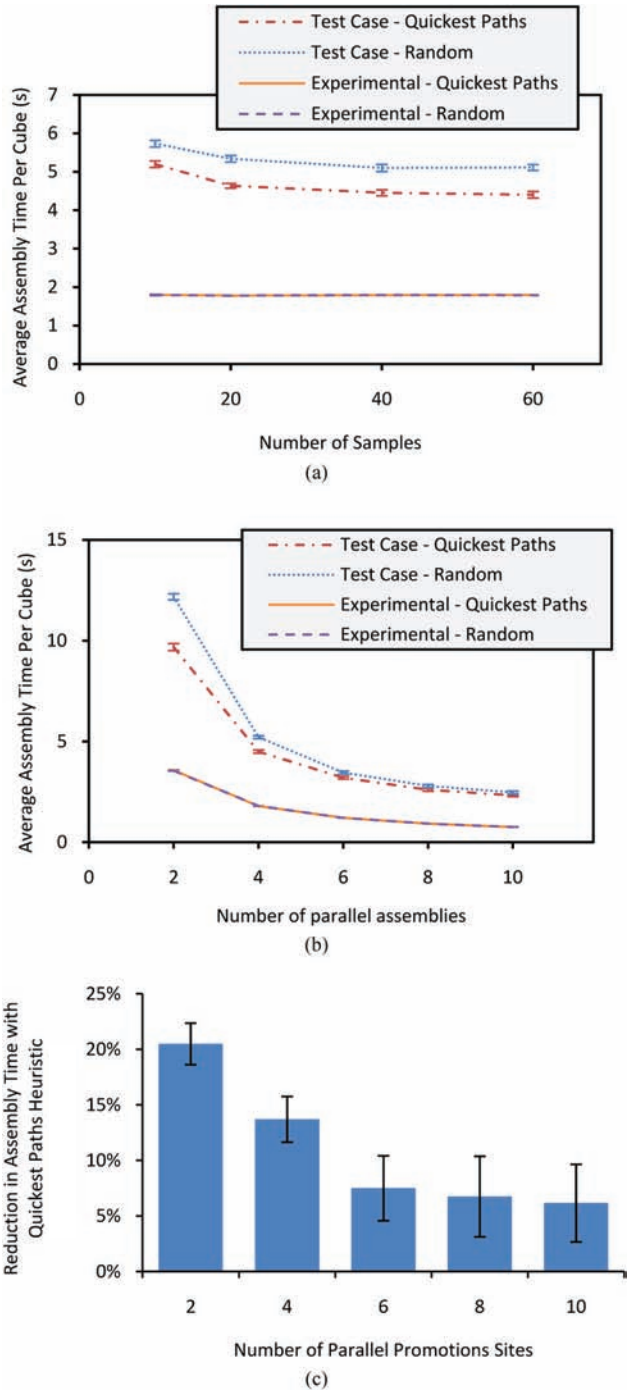


Fig. 11. Effect of number of samples and parallel promotion sites. (a) For the test assembly rates, increasing the number of samples decreased the assembly time per cube, up to a certain limit. The experimental assembly rates showed no change. (b) For both experimental and test cases, increasing the number of parallel promotion sites decreased the assembly time per cube. For the test case, the *Quickest Paths* heuristic again reduced assembly times, but showed diminishing returns with increasing parallel promotion sites. (c) Beyond four sites, the reduction in assembly time due to the *Quickest Paths* heuristic remained relatively constant at approximately 7%.

target structure. Due to the stochastic nature of the process, we would also like a confidence range for this estimate. For a serial assembly path, an estimate can be found simply by summing the expected assembly times for each cube addition along the path. However, we would like to be able to estimate the average time required to assemble a target structure while attracting at multiple locations in parallel, based on the information collected during sampling of possible assembly paths.

We estimate the expected parallel assembly rate by summing the sampled serial assembly rates. In order to determine a confidence interval for this estimate, we have to take two factors into account: the uncertainty in the expected assembly time and the variation due to the random, exponentially distributed arrival times of the modules. We deal with these two uncertainties separately in the next two sections.

6.5.1. Expected time uncertainty The uncertainty in the expected assembly time is due to the fact that this value is calculated from a set of samples of the structure, as described above in Algorithm 2. The parallel assembly rate is obtained by summing a number of serial assembly rates, each of which is estimated based on the number of random samples that end at the chosen locations. Assuming that these rates are each normally distributed and independent, we can calculate and sum their variations in order to obtain the standard deviation of the expected time. The indicated lines in Figure 12 demark the upper and lower bounds of the 95% confidence interval (i.e. 1.96 standard deviations) for the expected assembly time.

6.5.2. Stochastic arrival time distribution While we would expect a large number of repeated assembly experiments to approach the expected value estimated as above, it would in fact be quite surprising if this were the case for a single experiment. This is because each module is arriving at a random, exponentially distributed time. The distribution of the sum of a series of independent, exponentially distributed times is itself a gamma distribution. From the gamma distribution we can obtain 95% confidence bounds for the structure assembly time based on the expected values estimated as described in the previous section.

In Figure 12 we see the 95% confidence bounds from the gamma distribution for the upper and lower bounds of the expected value. For this particular run, we see that the simulated assembly time falls within these bounds for the majority of the assembly. Figure 12 also shows that a large part of the uncertainty in the overall structure assembly time comes from the random distribution of module arrival times. Nonetheless, the uncertainty in the expected value is also significant.

6.5.3. Evaluation of assembly time estimates We tested our assembly time predictions over the course of 120 runs

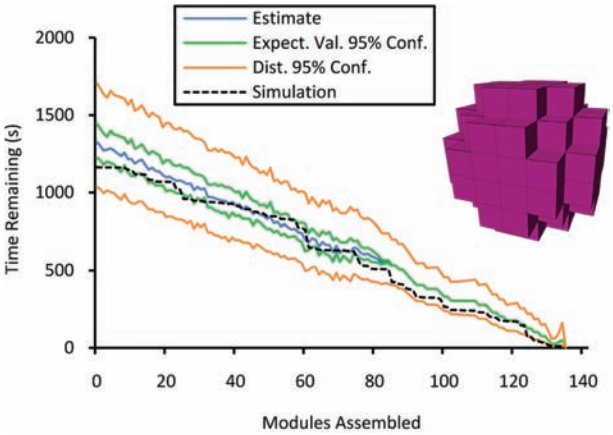


Fig. 12. Predicting stochastic assembly time. The remaining time required to assemble a target shape using the on-line stochastic assembly planner can be estimated based on the samples taken at each stage of assembly. Confidence bounds can be placed on these estimates by taking into account the uncertainty in the expected assembly time and stochastic distribution of the module arrival times. This plot shows the estimate and the 95% confidence range, taking first the expected time uncertainty, then both uncertainties, into account, in the simulated assembly of a 136-module spherical target structure. The actual simulated assembly time falls within these bounds.

of the 619-module hand target shape (as seen in Figure 9(a)) for both the experimental and test (expensive configuration 2) assembly rates described in Section 6.1. The results of these simulations can be seen in Figure 13. Part (a) shows the estimated assembly time remaining at each stage of assembly, as well as 95% confidence bounds and actual simulated assembly time for one particular run with the experimental values. (Note that the 95% confidence bounds on the estimate overlap the estimate itself due to the similarity of the three experimental expected cube arrival times.) Part (b) shows the same plot for the test values for which the second configuration was much more expensive than the others. Here we see a much larger range for the confidence bounds.

Figures 13(c) and (d) shows histograms of the results of similar simulations repeated 120 times for both assembly rate cases. The bars represent the number of simulations with total assembly times within the indicated ranges (20 s ranges in (c) and 100 s ranges in (d)). The bin containing the average of the expected times calculated in the 120 simulations is also indicated, as well as the average range of the 95% confidence bounds. In the first case (Figure 13(c)), 97.5% of the simulations assembly times fell within their 95% confidence range. In addition, the average simulated time (1,100 s) was very close to the average estimate (1,096 s). In the second case (Figure 13(d)), the average simulated time (3,964 s) was a fair bit lower than the average estimate (4,312 s) and only 76.7% of the simulation fell with the confidence range. Interestingly, if the expected value were corrected (i.e. if 348 s was subtracted from all predictions), then 93.3% of the simulations would fall within the 95% confidence range. This suggests that although the algorithm systematically over-predicted assembly times for the test case, it did give a reasonable confidence interval for the predictions.

One challenge in the calculation of the assembly time estimate is that our algorithm calculates a predetermined number of samples in order to choose the set of locations to attract modules (40 samples were used in these simulations). In addition, there is no way of specifying which locations are associated with each sample (since the paths are chosen randomly). Thus, any particular location may end up with an insufficient number of samples to give an accurate estimate of the uncertainty in the expected assembly time. This is a problem inherent with our sampling approach to be addressed in future work.

6.6. Computational effort and scaling

We studied the scaling of the simulated assembly time and the total planning computation time as a function of the number of modules for four classes of target structures (cubes, mirrored pyramids, heads, and hands, see Figure 14). Each data series contains five data points corresponding to five scaled versions of one of the classes of target structures pictured. Versions of each of the target structures were generated consisting of approximately 100, 200, 400, 800, and 1,600 modules. Each data point represents either the assembly (blue) or planning (red) time, averaged over 20 simulated assemblies of the indicated target structure. The planning computation time is the total time taken to run the entire assembly simulation on one core of an Intel Core 2 Quad Q6600 processor. For the assembly simulations, the experimental assembly rates found in Section 5 were used.

As expected, the simulated structure assembly time scales linearly with the number of modules, and no significant difference was found between the four classes of shapes. For all four classes of target structures, the assembly times increased at a rate of 1.7 s/cube, which is consistent with previous results. The assembly rates fit a linear curve with a coefficient of determination of 0.992.

As for the assembly planner, we can first predict some analytical bounds on its scaling. In the worst case, we expect the computation requirements of the assembly planner to scale with $O(kn^4)$, where n is the number of cubes in the structure and k is the number of samples performed at each step. This is because in the worst case the sequence generator (Algorithm 1) must visit every cube in the structure during the connectivity check, and in finding an accessible cube, for each disassembly step. Finally, Algorithm 2 calls Algorithm 1 k times for each cube in the target structure.

For a lower bound, we assume that Algorithm 1 finds an accessible cube and checks it for connectivity in a single step each time. In this case, Algorithm 1 scales with $O(n)$

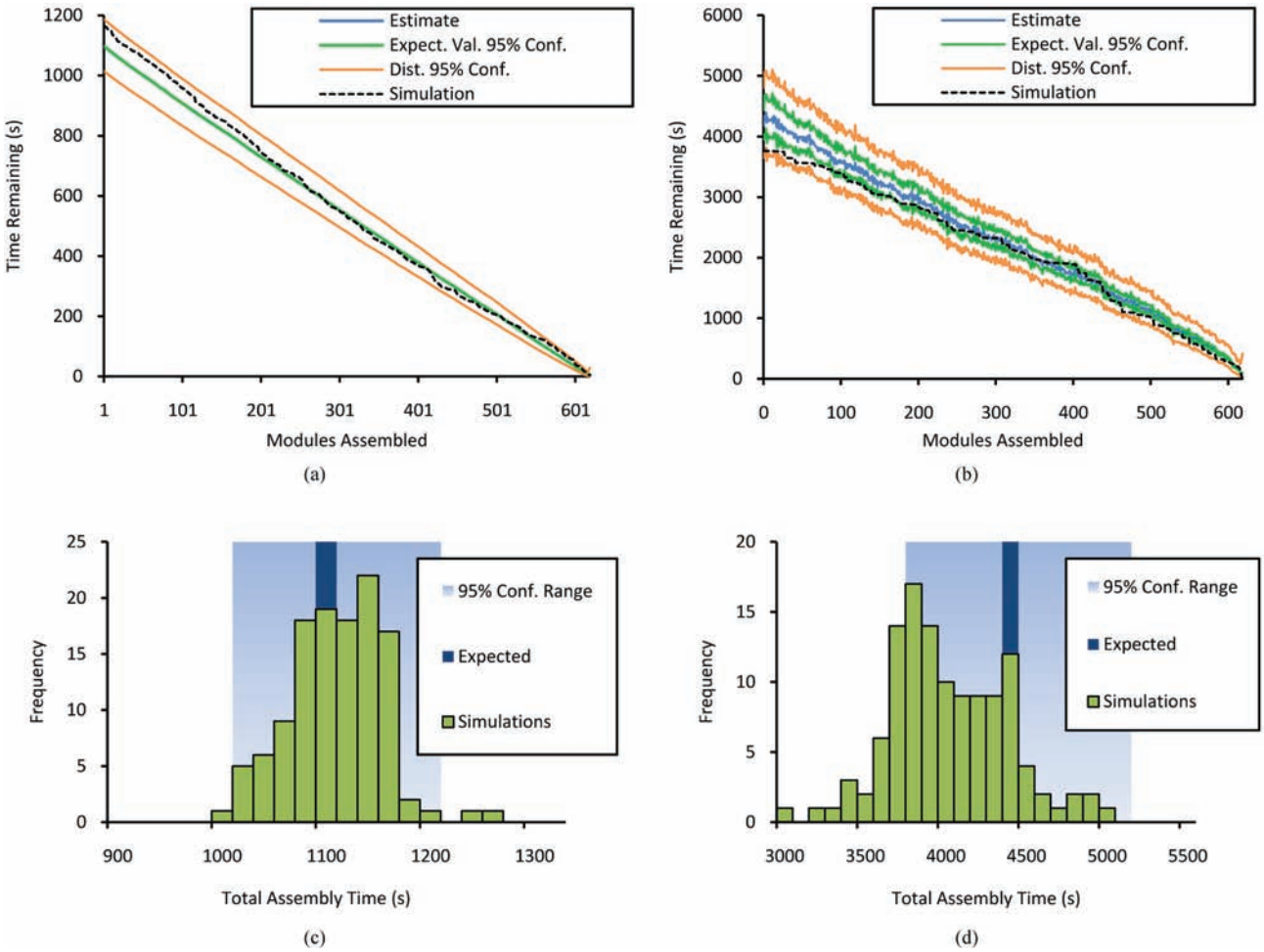


Fig. 13. Evaluation of assembly time prediction confidence bounds. (a) Plot of estimated and simulated time to assembly for the 619-module hand shape (see Figure 9(a)) using the experimental cube assembly rates described in Section 6.1. (b) Same as (a) except using the test assembly rates with an expensive two-outlet configuration. (c) Histogram of simulated total assembly times over 120 runs for the hand shape using the experimental cube assembly rates. Also indicated are the average expected assembly time and the average 95% confidence range. (d) Same as (c) except for the test assembly rates.

and thus the assembly planner scales with $O(kn^2)$. In practice, we would expect the time complexity of the assembly planner to scale somewhere between these extremes.

Looking at Figure 14, we find that the time required to compute the assembly paths on-line scaled differently depending on the class of target structures. However, the scaling was polynomial in each case, with an exponent value between those predicted above. The class of shapes that scaled the most favorably was the mirrored pyramids, with an exponent of 2.4. Following this were the cubes, heads, and hands, which scaled with exponents of 2.6, 2.8, and 2.9, respectively. The coefficients of determination for these power law fits were 0.991, 0.990, 0.989, and 0.999, respectively.

7. Discussion

The previous section highlights one of the challenges of the algorithm proposed in this paper. The cost of using a sample-based on-line algorithm that finds multiple parallel assembly paths that are guaranteed to result in error-free assembly is a high time complexity. For structures consisting of many modules, the assembly rate becomes limited by algorithm computation as opposed to physical assembly. Our results predict that this situation would be encountered with a target structure of greater than approximately 1,000 modules. This limits the applicability of our approach to larger systems. However, to date experimental stochastic assembly systems comprised of more than a few dozen modules have not been demonstrated. Thus we expect this approach to be viable for a variety of platforms.

Furthermore, aside from using faster processors or multi-threading, we foresee ways the experimental code could be optimized for speed. Firstly, the time complexity of the algorithm could potentially be reduced by choosing cubes to check for accessibility in a less naïve way (e.g. considering only surface cubes). In addition, for cases where the gains

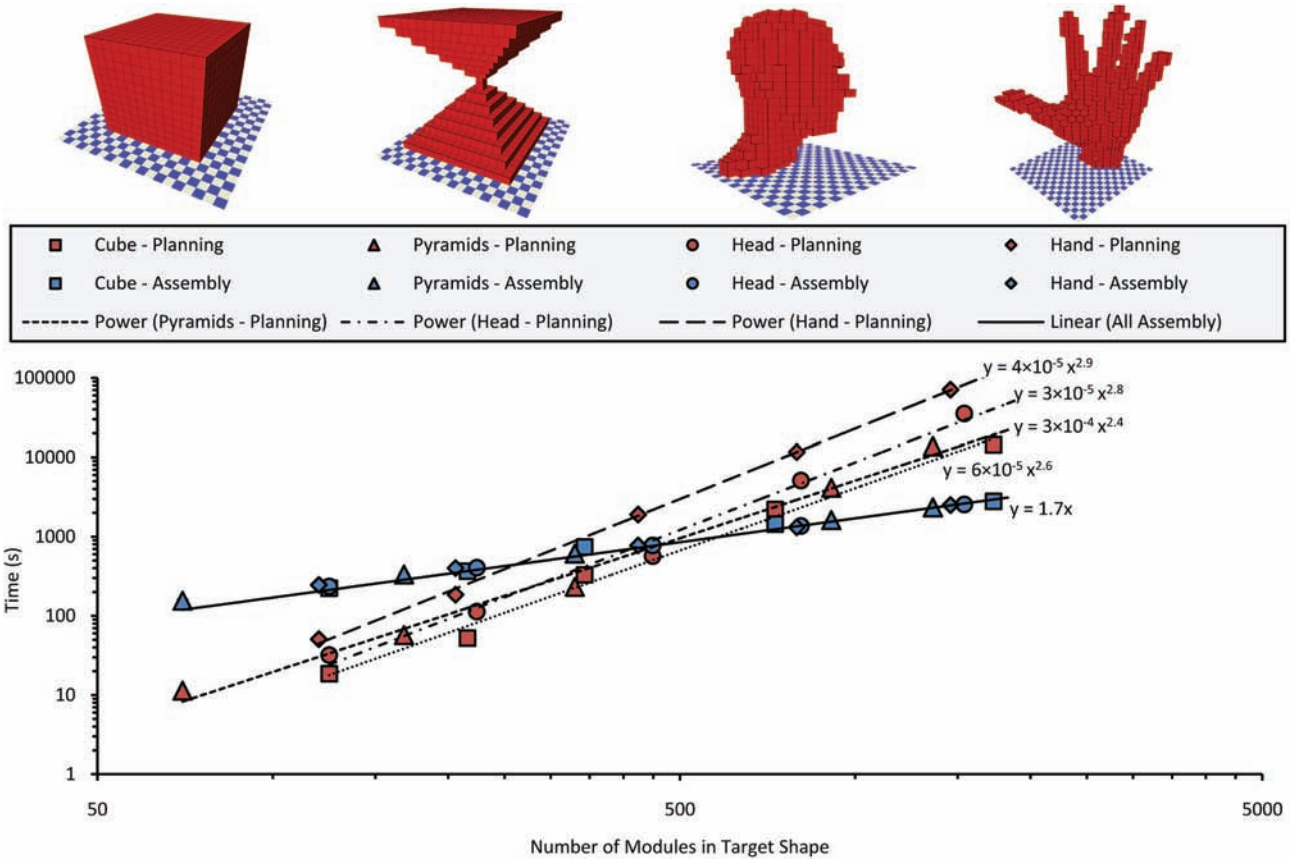


Fig. 14. Assembly planning algorithm scaling. This plot shows the scaling of the stochastic assembly time and the computation time required by the proposed on-line parallel assembly planner versus the number of modules in the target structure. These times were recorded from simulations of five different sizes of the four pictured classes of target structures (the largest size is pictured in each case). Each data point represents an average of 20 simulation runs. Axis scales are logarithmic. Linear (assembly data) and polynomial (planning data) fits are shown, with the corresponding equations indicated (color online only).

due to promotion site selection are small or non-existent (e.g. for the case shown in Figure 9(b)), the number of samples required could be greatly reduced by simply choosing the first N distinct promotion locations (where N is the maximum number of promotion locations). In addition, since our algorithm is re-sampling the same overall assembly graph over and over again, it may be possible to employ dynamic programming techniques to reduce the algorithmic complexity by re-using previous samples.

8. Conclusions

We have presented an approach that provably finds assembly sequences for any finite structure (Algorithm 1). Using this algorithm repeatedly to sample assembly sequences, we have developed an approach to parallel assembly that is applicable to a variety of stochastic robotic systems (Algorithm 2). The on-line nature of this approach makes it adaptable to changing inputs or environments, and allows it to predict the assembly time remaining. We have demonstrated the application of this approach to the assembly of structures with various topologies in simulation. In addition, for the case where the number of assembly promotion sites is limited, we have investigated various heuristics for minimizing assembly times. We have also examined the trade-offs of varying primary algorithm parameters (the number of samples and the number of parallel promotion sites). We have also tested the approach's ability to accurately predict assembly times and give confidence bounds on these predictions. Finally, our scaling analysis suggests that this algorithm scales to the assembly of at least 1,000 modules, and we have suggested ways this scaling may be improved.

Additional future work could investigate scaffolding by allowing modules to be removed, as well as added, during assembly. Error identification and correction would also certainly be required for a practical stochastic assembly system. It would also be interesting to investigate adding further constraints during assembly, such as distributing promotion sites over the surface of the structure when possible, and accounting for the structural properties of substructures during assembly. Taking more of the local configuration into account when calculating assembly rates would

most likely improve the accuracy of assembly simulations. In addition, using a hierarchical approach, where the proposed algorithm is used to plan the assembly of subsystems, which are then united using another approach, could potentially help solve the scaling problem. Finally, we await the opportunity to test this stochastic assembly approach on an experimental system.

Acknowledgements

The authors would like to thank the members of the Computational Synthesis Lab at Cornell University for their advice throughout this work. In particular, we would like to thank Michael Schmidt, Aaron Lenfestey, Jon Hiller, Jonas Neubert, and Diana Chang for their invaluable suggestions and assistance.

Funding

This work was supported in part by the Defense Advanced Research Projects Agency Defense Science Office Programmable Matter Program (grant number W911NF-08-1-0140), and the U.S. National Science Foundation's Office of Emerging Frontiers in Research and Innovation (grant number 0735953). The work of MT Tolley was supported by the Natural Sciences and Engineering Research Council of Canada through the Postgraduate Scholarship Program.

Conflict of interest

The authors declare that they have no conflicts of interest.

References

Ayanian N, White PJ, Halaz A, Yim M and Kumar V (2008) Stochastic control for self-assembly of XBots. In: *Proceedings of ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, New York, pp. 1169–1176.

Detweiler C, Vona M, Yoon Y, Yun S.-K and Rus, D (2007) Self-assembling mobile linkages. *IEEE Robotics and Automation Magazine* 14: 45–55.

Everist J, Mogharei K, Suri H, Ranasignhe NK, Will P and Shen W-M (2004) A system for in-space assembly. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sendai, Japan, pp. 2356–2361.

Fitch R, Butler Z and Rus D (2003) Reconfiguration planning for heterogeneous self-reconfiguring robots. In: *Intelligent Robots and Systems*, Las Vegas, NV, pp. 2460–2467.

Gibson MA and Bruck J (2000) Efficient exact stochastic simulation of chemical systems with many species and many channels. *Physical Chemistry A* 104: 1876–1889.

Gillespie DT (1977) Exact stochastic simulation of coupled chemical reactions. *Physical Chemistry* 81: 2340–2361.

Gilpin K, Kotay K, Rus D and Vasilescu I (2008) Miche: modular shape formation by self-disassembly. *The International Journal of Robotics Research* 27: 345–372.

Griffith S, Goldwater D and Jacobson JM (2005) Robotics: self-replication from random parts. *Nature* 437: 636.

Hjelle DA and Lipson H (2009) A robotically reconfigurable truss. In: *Proceedings of ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots,* London.

Jones C and Mataric M (2004) Automatic synthesis of communication-based coordinated multi-robot systems. In: *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, pp. 381–387.

Klavins E, Ghrist R and Lipsky D (2006) A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control* 51: 949–962.

Krishnan M, Tolley MT, Lipson H and Erickson D (2009) Hydrodynamically tunable affinities for fluidic assembly. *Langmuir* 25: 3769–3774.

LaValle S and Kuffner JJ (1999) Randomized kinodynamic planning. In: *Proceedings of the International Conference on Robotics and Automation*, Detroit, MI, pp. 473–479.

Lerman K, Jones C, Galstyan A and Mataric AJ (2006) Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research* 25: 225–241.

Lobo D, Hjelle DA and Lipson H (2009) Reconfiguration algorithms for robotically manipulatable structures. In: *Proceedings of the ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots*, London.

Murata S and Kurokawa H (2007) Self-reconfigurable robots: shape-changing cellular robots can exceed conventional robot flexibility. *IEEE Robotics and Automation Magazine* 14: 71–78.

Nagpal R, Shrobe H and Bachrach J (2003) Organizing a global coordinate system from local information on an ad hoc sensor network. In: *Proceedings of the International Workshop on Information Processing in Sensor Networks*, Palo Alto, CA, pp. 333–348.

Napp N, Burden S and Klavins E (2006) The statistical dynamics of programmed self-assembly. In: *Proceedings of the IEEE International Conference of Robotics and Automation,* Orlando, FL.

Rus D and Vona M (2001) Crystalline robots: self-reconfiguration with compressible unit-modules. *Autonomous Robots* 10: 107–124.

Sacerdoti E (1975) The nonlinear nature of plans. In: *Proceedings of the Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, p. 206.

Terada Y and Murata S (2004) Automatic assembly system for a largescale modular structure - hardware design of module and assembler robot.In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, pp. 2349–2355.

Tolley MT, Kalontarov M, Neubert J, Erickson D and Lipson H (2010) Stochastic modular robotic systems: a study of fluidic assembly strategies. *IEEE Transactions on Robotics* 26: 518–530.

Tolley MT, Krishnan M, Erickson D and Lipson H (2008) Dynamically programmable fluidic assembly. *Applied Physics Letters* 93: 254105-1-3.

Tomita K, Murata S, Kurokawa H, Yoshida E and Kokaji S (1999) Self-assembly and self-repair method for a distributed mechanical system. *IEEE Transactions on Robotics and Automation* 15: 1035–1045.

Varshavskaya P, Kaelbling LP and Rus D (2008) Automated design of adaptive controllers for modular robots using reinforcement learning. *The International Journal of Robotics Research* 27: 505–526.

Werfel J, Bar-Yam Y, Rus D and Nagpal R (2006) Distributed construction by mobile robots with enhanced building blocks. In: *Proceedings of International Conference on Robotics and Automation*, Orlando, FL, pp. 2787–2794.

Werfel J and Nagpal R (2008) Three-dimensional construction with mobile robots and modular blocks. *The International Journal of Robotics Research* 27: 463–479.

White P, Zykov V, Bongard J and Lipson H (2005) Three dimensional stochastic reconfiguration of modular robots. In: *Proceedings of Robotics Science and Systems*, Cambridge, pp. 161–168.

White PJ, Kopanski K and Lipson H (2004) Stochastic self-reconfigurable cellular robotics. In: *Proceeding of the IEEE International Conference on Robotics and Automation*, vol. 3, New Orleans, LA, pp. 2888–2893.

Yim M, Shen W-M, Salemi B, Rus D, Moll M, Lipson H, et al. (2007). Modular self-reconfigurable robotic systems. *IEEE Robotics and Automation Magazine* 14: 43–52.

Zykov V and Lipson H (2007) Experiment design for stochastic three-dimensional reconfiguration of modular robots. In: *Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Self-Reconfigurable Robotics Workshop,* San Diego, CA.

Appendix: index to multimedia extensions

The multimedia extension page is found at http://www.ijrr.org

Table of Multimedia Extensions

Extension	Type	Description
1	Video	Narrated animations illustrating the proposed stochastic assembly-planning algorithm