2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Decentralized Robotic Assembly with Physical Ordering and Timing Constraints

T. Ryan Schoen and Daniela Rus

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
rschoen@mit.edu, rus@csail.mit.edu

***Abstract*— Our prior work presented a system for decentralized robotic assembly[1]: given a team of robots, a cache of components, and a desired structure specified as a blueprint, the algorithm computes a sequence of part deliveries and assembly steps to achieve the desired structure, while considering physical dependencies and reachability constraints for the goal structure. In this paper we introduce a new algorithm that extends our prior result to incorporate the duration of each assembly operation. We also extend the algorithm to be adaptive to the availability of parts. When a part is not available, the assembly sequence is recomputed. The algorithms are provably convergent and their execution does not depend on the size of the robot team. We implement the algorithms using a team of four youBot robots that can (1) locate and identify parts; (2) use communication to achieve coordinated hand-off of parts; and (3) create complex log-cabin style structures.**

I. Introduction

Distributed robotic assembly is increasingly becoming a focus of robotic efforts[2], as the applications span varied industries. There are many properties we would like to see in a robotic assembly system to maximize its usability in practical situations. It should be completely decentralized, allowing robots to operate independently with only local information. This allows scaling of the system to an arbitrary number of robots or size of assembly. The system should be robust to failure, such that a point failure in one or more robots does not prevent the overall structure from reaching completion. Efficiency should be maximized, utilizing the robots and parallelizing tasks to the greatest extent possible. And finally, as the robots are constructing a physical structure, the system should be aware of the constraints and physics of the world in which they are interacting. Thus they can ensure a correct construction and integrate this knowledge to maximize efficiency.

Our previous work in robotic assembly has focused on the following aspects. We developed algorithms to equally split work between robots for parallelism, and controllers to enable the robots to carry out the construction[3]. These algorithms and controllers were fully decentralized and scalable. The system was robust to failure among robots[4]. We implemented these algorithms on a system to demonstrate their practical properties[5]. However, the system did not regard the physical constraints of the structure while scheduling, one of the key traits of a complete robotic assembly platform.

In this work, we present new algorithms for a decentralized scheduling approach to complex assembly that models the time required by each operation and is adaptive to part availability. The new algorithms consider the physical constraints of an assembly and utilize them to tend toward efficiency and parallelism.

We also redesign our robotic assembly system in this work, and implement it on a new hardware and software platform. Each robot can locate and grasp a part from a part cache, hand that part to another robot (from grasped configuration to grasped configuration without the need to put the part down) and add the part to the assembly. We then use this platform to investigate the full assembly process with our construction ordering algorithms.

The robotic platform consists of four KUKA youBots, featuring a holonomic base and a five degree-of-freedom arm. We use a Vicon motion capture system for localization and standard Wifi for communication. The algorithms and software architecture are implemented on Robot Operating System (ROS), written in C++ and Python. The schematics for the target structures are provided to the robots in a YAML blueprint defined in terms of the component parts of the structure.

A. Related work

Much of our prior work has addressed three-dimensional assembly[6], [7], [8]. Specific to distributed assembly, our previous work investigated and developed theoretical algorithms for equal-mass partitioning, coverage, multi-robot coordination, and parallel assembly[3], [4], [5], [9], [10]. Our recent work extends these algorithms to integrate physical constraints[1], but does not consider these constraints while scheduling.

Several others have investigated similar systems and their desirable properties. Yasuda et al. described a multi-agent system with robust error tolerance[11]. Van Brussel et al. investigated multi-agent systems from the perspective of distribution in an ever-changing environment[12]. Stochastic policies for parallel task allocation in robotic swarms were investigated by [13]. Stochastic algorithms for robotic construction with dependency on raw materials were analyzed by Matthey[14]. [15] developed methods for evaluating the complexity of structures, as it applies to their distributed robotic construction. Three-dimensional construction with consideration to physical constraints such as gravity and stacking was achieved by [16], [17], and [18], among others.

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

II. Review of Problem Formulation: Coordinated Robotic Construction with Constraints

This section summarizes the decentralized construction model from our previous work[1].

We are given a team of robots, divided into two types: those which deliver parts to the assembly area, and those which assemble the structure. The robots can communicate locally with other robots within their communication range. They are all provided a specification of the assembly to be constructed, including a specification of its component parts. The blueprint also contains a representation of two directed acyclic graphs (DAGs) G_p and G_r which define the physical dependency and reachability constraints of the structure.

We define a physical dependency between two parts if the one part cannot be stably placed or assembled until the other has been, regardless of the state of any other part. Similarly, we define reachability between two parts if a robot at the location of one part can reach the other part, regardless of the state of any other part. With these dependencies we can form a topological sort of the parts required for assembly. This provides a loose assembly ordering, but does not address the issue of ordering for efficiency and parallelism.

This problem formulation allows the construction of any arbitrary structure, but for simplicity in exposition we will use two sample structures: a log cabin design and a staircase as described in Section V.

At a high level, the assembly and delivery algorithms we employ are identical to those employed by Bolger et al[5]. The main algorithm is reproduced here in Algorithm 1. After a random deployment of assembly robots across the construction site, delivery robots begin to fetch parts from a source location and bring them to the assembly robot with the most work to complete, or the most *demanding mass*. The assembly robots then use these delivered parts to construct the target structure. In this work we introduce a new hardware and software platform as described in Section V, and employ the novel partitioning and ordering algorithms described below. These algorithms fundamentally alter the distribution of demanding mass among the robots, resulting in a prioritization of assembly parts not found in prior work.

Algorithm 1 Construction Algorithm

1: Deploy the assembly robots
2: Place the assembly robots at optimal task locations
3: **repeat**
4: **delivery robots:** carry source components to the assembly robots
5: **assembly robots:** assemble the delivered components
6: **until** task completed *or* out of parts

III. Adaptive Construction with Part Unavailability and Varying Operation Times

Previous work has assumed that the supply of parts is well-stocked, and that each assembly task requires roughly the same amount of time. In practice, these assumptions rarely hold. In this section we present a decentralized adaptive algorithm where the structure to be created requires different types of assembly operations and each assembly operation is parameterized by the time required to execute it. We also consider the availability of the parts required for the current assembly step and extend the algorithm to incorporate adaptation to part availability. If a part is not available, the construction algorithm recomputes the assembly schedule to re-focus the operations where it is possible to make progress.

A. Part Supply

In a practical assembly operation with heterogenous parts, there is not an infinite supply of materials. During the assembly process the supply of a particular part may run out and be resupplied later. A robust assembly algorithm should adapt to this change.

Let the function $n(\lambda)$ represent the number of parts available of type λ. We assume that all robots have information about the number and type of parts available, although for our purposes it is sufficient to represent $n(\lambda) \in \{0, 1\}$ as the presence or lack of parts of type λ.

The first modification to our algorithm ensures that a part which we lack is not considered "placeable". We introduce a boolean variable to represent this information, and mark a part as unplaceable if it is true.

$$\xi_s(v) = (n(\lambda_v) = 0) \tag{1}$$

The two DAGs described in Section II are defined as $G_p = (V, E_p)$ and $G_r = (V, E_r)$, where the nodes $v \in V$ represent the discrete assembly operations to be completed and the edges E_p and E_r represent the physical and reachability constraints respectively. We introduce two algorithms: one to be run when a robot receives communication that a supply of a certain part type has been extinguished, and another when it receives communication that a part type has been replenished. These two algorithms are found in Algorithms 2 and 3, respectively.

Algorithm 2 Part has been extinguished

1: Receive communication that $n(\lambda) = 0$
2: $E_p^\lambda \leftarrow \emptyset$
3: $E_p' \leftarrow E_p$
4: **for** $(v_i, v_j) \in E_p : \lambda_{v_j} = \lambda$ **do**
5: $E_p^\lambda \leftarrow E_p^\lambda \cup \{(v_i, v_j)\}$
6: $E_p' \leftarrow E_p' \backslash \{(v_i, v_j)\}$
7: **end for**
8: $E_p \leftarrow E_p'$

Algorithm 3 Part has been replenished

1: Receive communication that $n(\lambda) > 0$
2: $E_p \leftarrow E_p \cup E_p^\lambda$

The first improvement from this modification ensures that a part is not considered placeable if its supply has run

out, because an assembly robot will not assign positive demanding mass to those parts. This prevents delivery robots from seeking that part.

Second, by removing the physical dependency of extinguished parts from the parts they depend on, the assembly algorithm now weights those depended-on parts less. This is desired, because part types that are currently extinguished do not contribute to the pool of available work. Placing physical dependencies of the lacking parts does not free up more parts for the assembly robots to place. By pretending that the parts no longer exist in the assembly blueprint, we ensure we that we maintain efficient ordering construction.

In this algorithm we did not modify the reachability graph, G_r. This is done so that even though we are ignoring the extinguished parts in the dependency graph, we do not place parts that would block the placement of the extinguished part once the part supply has been replenished.

B. Assembly Time

Our previous algorithms have assumed that all assembly operations take the same amount of time to accomplish, which is rarely true in practice. For example, one can imagine an assembly where one task is twice as valuable as another to complete in order to maximize parallelism and efficiency; our prior algorithms would choose the former task to complete first. However, if the first task takes three times as long to complete than the other, then the second task is actually more desirable to complete first. It will make additional work available sooner.

Once we introduce the concept of assembly time, we are no longer interested in the ability of task completions to make work available; instead, the quantity of interest is a task's ability to make additional work available divided by the amount of time it takes to complete that task. Mathematically, we introduce this into the algorithm by altering the scoring function described in our previous work. It has the critical property that it produces higher priority for those parts that would make the most work available (tending toward parallelism) and is in the hardest-to-reach area first (tending toward efficiency). It now takes the form:

$$score(f(\cdot), X) = \sum_{x \in X} \left(2^{\frac{f(x)}{\tau_x}}\right)^{-c_X} \tag{2}$$

where τ_x represents the amount of time required to complete task x and the function $f(x)$ represents the number of other nodes that will be affected by completing a task (for example, by fulfilling physical dependencies). By dividing $f(x)$ by the time required to complete that task, we now weight assembly tasks according to their actual value to the process.

Theorem 1: The controller modified by timing constraints will converge to a complete structure if possible.

Proof: The original controller converges to a complete structure if possible, per the proof in [1]. This proof, which is not duplicated here for space reasons, is based on the structure of the dependency DAGs and the mass function, both of which are independent of the exponent $\frac{f(x)}{\tau_x}$. Therefore the modification of this parameter does not alter convergence, and the controller will converge to a complete structure if possible. ■

IV. Simulation Results

To test the effectiveness of our algorithms, we modified the assembly simulator used in [1] to take part supply into account. In this simulation, the wireframe of a plane must be constructed prior to placement of the panels that make up the plane's body. By artificially modifying the supply of these parts, we can observe the effects of our algorithmic changes.

We ran the simulation on the airplane assembly seen in Figure 1 using six assembly robots and six delivery robots. At $t = 20$ the supply of plane fuselage panels is extinguished. At $t = 80$ the supply is replenished so that the assembly can be fully constructed. The simulation was run twenty times: ten times using the original algorithm, and ten times using the modified part supply algorithm. The results from each set of ten were averaged together.

A graph of average demanding mass over time is found in Figure 2. The vertical lines at $t = 20$ and $t = 80$ show when the supply of plane panels was extinguished and resupplied, respectively. As expected, both algorithms perform roughly the same until $t = 80$. The difference up until this point is not the amount of work being done, but that with the modified algorithm the robots are more intelligently choosing *which* work to do in order to efficiently parallelize the work once the part supply is replenished.

It follows that the behavior diverges after $t = 80$. The original algorithm produces a slightly larger spike in available work - this is expected, since the unmodified weighting function would have caused the dependencies of plane panels to be assembled. Thus, when panels are resupplied there is a large and immediate need for them. However, under the original algorithm, there are undiscovered bottlenecks in the assembly process that have not been addressed. This results in a much longer overall completion time.

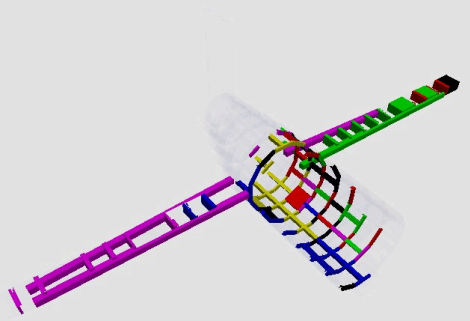
In contrast, the modified algorithm weighted the dependencies of the plane panels low (since they did not free up additional work to complete), and instead focused on other sources of bottlenecks. This leaves a large amount of available, parallel work to be completed. Once the plane panels are resupplied, the robots can place the panels' dependencies and the panels themselves. The modified algorithm finishes the construction task much faster than the original algorithm.

V. Experimental System

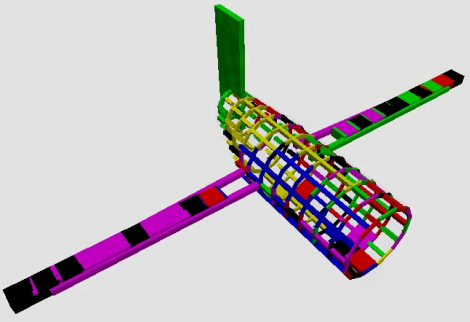
A. Mobile manipulator

The platform on which we have chosen to implement our algorithms is a team of four KUKA youBots. The youBot, seen in Figure 3, consists of a holonomic base capable of omnidirectional movement and a five degree-of-freedom arm with two-finger gripper[19]. The robots are equipped with an onboard PC running Ubuntu Linux. The mini ITX PC board also contains embedded Wifi to allow the robots to

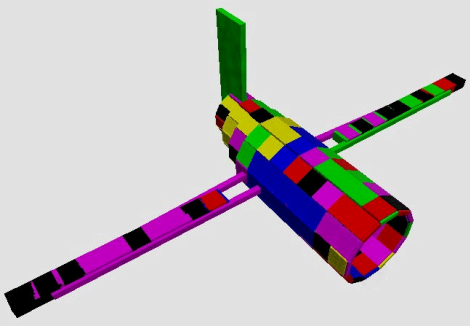
communicate with one another, although we have augmented them with Netgear WNCE2001 Wifi adapters for increased communication integrity.



(a) At $t = 20$ the fuselage panels have run out. Much of the structure is left to complete.



(b) At $t = 80$, the fuselage panels are resupplied. The assembly robots have constructed much of the framework of the fuselage, but were unable to place any fuselage panels in the past 60 timesteps.



(c) The plane has been fully assembled at $t = 102$ with the resupplied fuselage panels.

Fig. 1. The plane assembly used in simulation contains a fuselage, two wings, and a tail section, each of which is composed of many individual parts. The parts are color-coded to indicate which of the six assembly robots placed each part (e.g., red parts were placed by robot 1, green by robot 2). There are a variety of structural dependencies between parts, making the construction order complex. The plane is shown at (a) $t = 20$, (b) $t = 80$, and (c) completion at $t = 102$.

Our previous platform allowed for easy navigation and the manipulation of our custom-built objects at ground level[5]. However, it suffered from a lack of precision and a relatively small workspace that inhibited efforts to build structures in three dimensions.

In contrast, the KUKA youBot enables a much larger workspace with more degrees of freedom. The minimum

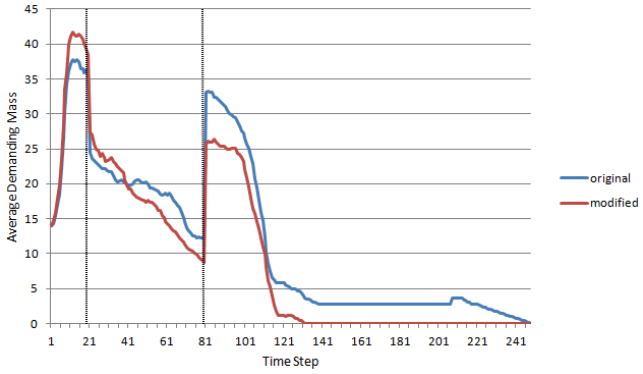


Fig. 2. The average demanding mass over time of ten simulations using the original algorithm (blue) and ten simulations using the modified algorithm (red). At t=20 the supply of plane panels is extinguished; at t=80 the supply is replenished.

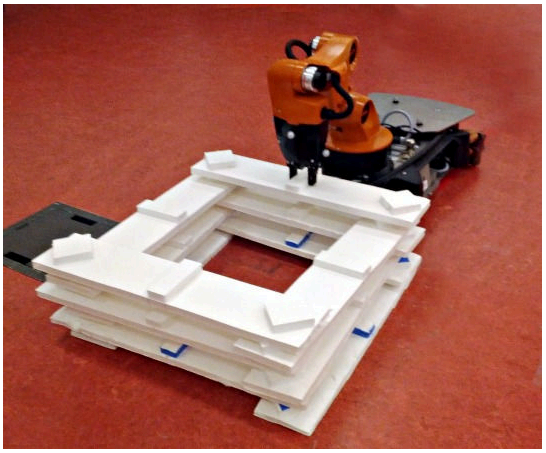


Fig. 3. An assembly robot places the final part on the three-dimensional tower. The tower is composed of six layers of the log cabin construction, or three of the simple squares from Section VII. This tower is the result of trial #1 from Table II.

speed of the holonomic base is small enough to allow for minute adjustments. Because of these factors, in this work we present a three-dimensional structure that would have been impossible to construct using our previous platform.

B. Localization

Localization for the youBots is provided by a 12-camera Vicon motion capture system, which can track position and orientation to millimeter and milliradian precision respectively. Retroreflective markers allow the system to identify robots and objects in the workspace.

In our experiments, the robotic base and manipulator were separately marked. The base was tracked for navigation, collision avoidance, and rough navigation toward a goal location. The arm was tracked for fine position adjustments and precise manipulation.

The poses of both the bases and arms are broadcast wirelessly to the robots at 10Hz using the tf interface for ROS, as described below.

C. Software and communication

The software architecture runs within the Robot Operating System (ROS). There are several nodes, depicted in Figure 4, that run simultaneously. At the lowest level, there are hardware-specific nodes to control the robotic arm and base through ROS wrappers for the youBot driver. These in turn are given commands by the planner, which directs the overall flow of the assembly process. The planner is in constant communication with the blueprint node, which maintains the state of the assembly process and the goal structure. The blueprint node coordinates with the partitioner node, where the heart of the algorithm exists. The partitioner ensures an efficient assembly order evenly split among the robots. Finally a Vicon node interacts with the Vicon motion capture system to provide position information to the partitioner and planner, which use the data to spatially partition the work and issue execution commands appropriately.

Our system takes advantage of the distribution and communication infrastructure in ROS. All nodes are run in a decentralized manner on the appropriate robot, with the exception of the Vicon system which is necessarily centralized. Communication is performed through ROS channels or "topics". All nodes are designed to function with an arbitrary number of robots, although the experiments described here will only focus on two or four in order to demonstrate the specifics of the system.

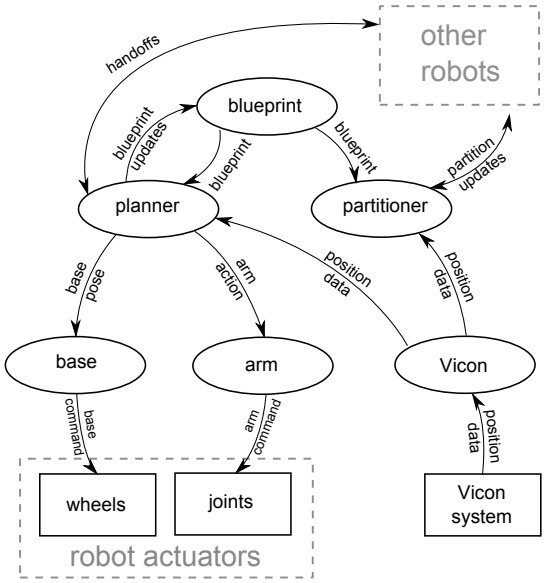


Fig. 4. System architecture and information flow. Each oval represents a separate ROS node, and the arrows indicate messages being passed between nodes (or in some cases, between robots).

D. Blueprints and parts

Structures are specified using the YAML markup language. A blueprint file is a list of parts, each of which contains a unique identifier, a pose in the target structure, and any dependencies the part may have or provide. Using this simple but robust description, an arbitrarily complex structure can be specified.

As a simplified structure, we decided to build squares in an approach similar to how a log cabin is constructed. Two parallel parts are stacked on alternating sides. Six layers of this structure are shown in Figure 3. The parts are constructed out of lightweight foam, and consist of a long flat section with two diamond-shaped supports on either end and a raised gripping area in the center.

To demonstrate our part supply algorithm, we will make use of styrofoam cubes to build a staircase. For part heterogeneity, we have split the cubes into two groups and color-coded them accordingly. The red cubes represent the finished "tops" of the staircase, whereas the blue cubes represent the unfinished "foundation" of the staircase, perhaps made out of concrete in a real assembly process. The assembled staircase has three stairs - one red cube, one red cube on top of one blue cube, and one red cube on top of two blue cubes.

VI. Implementation

A. Navigation

A motion planner described in [20] is employed for rough navigation. It uses a combination of a grid-based global planner and an equivalence class-based local planner to calculate a smooth and safe path to the goal. A static map assigns high cost to the part cache and target construction area, such that the robots only approach them as needed to retrieve or place parts.

For any navigation task, the robots switch to a second controller upon reaching the general vicinity of the goal. This controller uses simple proportional feedback control on the base velocity given the position of the arm in order to precisely position the robotic manipulator for the next step in the task.

B. Manipulation

In our previous work, we took advantage of smart parts and a gripper with an infrared sensor in order to locate and accurately grasp component parts[5]. To better generalize to all robotic assembly tasks, we have removed these constraints from our new system. In these experiments neither the parts nor the manipulators were equipped with any sensing or vision. Additionally, the parts were not marked with the retroreflective markers. Instead, the robots relied on consistent part source location.

To assist in manipulation, we added small sandpaper discs on the inside of the youBot fingers. This prevented the parts from slipping or changing orientation. We also added these sandpaper discs onto the colored cubes used in the part supply algorithm, to further prevent slipping.

C. Handoffs

An important innovation of the manipulation system that is enabled by the youBot hardware is communication-based direct handoff for transferring parts between robots without having to place the parts on the ground as in [1]. The intuition behind the handoff algorithm is as follows. The

delivery robot moves to the cell of the assembly robot for part handoff. At this location, the robot uses its pose and model of its own hardware to calculate the location of the part, and the grasping location on the part. This information is broadcast to the assembly robot, which navigates to the same cell and calculates its position and the configuration of its manipulator so as to to position its gripper in the handoff location. Next, the assembly robot grasps the part and then communicates this state to the delivery robot which then releases the part. Both robots then move away from each other and continue their respective tasks. An image of part handoff just before the delivery robot releases the part can be seen in Figure 5.

This direct handoff capability increases the speed and accuracy of the assembly operation. The solution is robust and is explored further in Section X.

D. Ordering

As described in Section II, our ordering relies on two DAGs G_p and G_r represented in the assembly blueprint. These respectively indicate the physical dependencies and reachability constraints of the target assembly. Given the structure of these graphs and a scoring function, we can weight individual parts by their contribution to the parallelism and efficiency of the overall task[1]. Assembly robots therefore choose the parts with the largest weight to assemble next, ensuring that the robots are greedily opening up the most future work to be done.

To save complexity and time over trials, the ordering of parts in the assembly process is calculated prior to execution. In practice this could be performed either offline or online, depending on the requirements of the task and the volatility of the assembly blueprint. The algorithms and approaches are the same regardless of the choice.

E. Delivery

After retrieving a part from the source, a delivery robot begins listening to broadcasts coming from assembly robots. These broadcasts contain each assembly robot's demanding mass for each part type. The delivery robot chooses the assembly robot with the highest demanding mass for the part type it has retrieved, and moves toward that robot.

Once the delivery robot is within appropriate range of the assembly robot, it hands the part to the assembly robot (Figure 5). Again, since the parts and grippers are not equipped with any sensors, both robots must be precise in their handoff.

F. Assembly

Finally, assembly is performed using the same precise techniques for a stable, accurate placement of the individual parts. Assembly repeats until the structure is complete or parts are no longer delivered via the delivery robots.

G. Robust Fault Recovery

Our algorithms and implementation allowed for robust recovery in the face of single- or multi-robot failures. Several

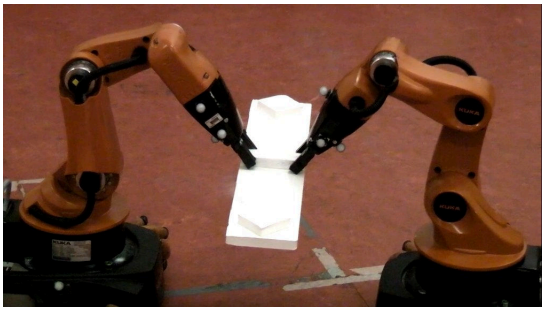


Fig. 5. Image of a delivery robot performing a delivery. Since the robots do not have vision and the assembly parts are not tracked by the Vicon system, the placement and communication of part coordinates must be precise.

Trial	**Runtime** (M:SS)	**Assembly utilization**	**Delivery utilization**	**Success** (Y/N)
1	6:04	83.4%	64.5%	Y
2	5:59	86.5%	65.9%	Y
3	6:09	80.7%	64.6%	Y
4	6:22	86.5%	63.7%	Y
5	6:24	87.8%	63.6%	Y
6	5:58	86.8%	63.7%	Y
7	6:05	88.3%	65.6%	Y
8	6:20	87.0%	66.2%	N
9	6:12	87.6%	64.4%	Y
10	6:17	85.9%	65.1%	Y
Avg	6:11	86.1%	64.7%	90%

TABLE I
SUMMARY OF ROBOT ASSEMBLY TRIALS FOR A SQUARE

times during demonstrations we experienced a failure of one or more robots; we were able to remove that robot from the assembly floor and continue the demonstration to completion.

This type of failure fortunately or unfortunately did not occur during the experiments that we describe. In order to highlight this aspect of our algorithms, we intend to explore experiments with artificially induced failures in the future.

VII. TWO-DIMENSIONAL EXPERIMENTAL RESULTS

A. Overall results

Ten full assembly trials were attempted with a blueprint of a simple square structure (two layers of the structure found in Figure 3), using just two robots. Each trial consisted of four delivery-assembly iterations. Each iteration involved a delivery robot retrieving a part from the source, delivering it to the assembly robot, and transmitting its location to the assembly robot; the assembly robot in turn retrieves the part from the broadcast location, moves to the assembly location, and places the part on the assembly. There were therefore many possible points of failure, especially given that the parts were retrieved and manipulated without vision.

The results are summarized in Table I. Assembly and delivery utilization is defined as the percent of time that the assembly or delivery robot, respectively, was busy with a task as opposed to waiting.

Over all ten trials, there was only one failure. This occurred when the assembly robot dropped a part after

retrieving it from delivery but before placement on the final structure. This was likely due to a low battery which caused there to be insufficient force in the gripper; the battery was replaced for the last two trials and no further issues were seen.

In the other nine trials, there were no significant failures and all structures were completed without intervention.

B. Runtime and efficiency

The average runtime over the trials was 6 minutes and 11 seconds, with a standard deviation of 9.6 seconds. An activity log for a typical run can be found in Figure 6. The solid bars indicate when each robot was busy with a task, and the lack of a bar indicates that the robot was waiting. The chart shows that the assembly robot was busy for nearly the entire assembly process, whereas the delivery robot was busy for significantly less time. Indeed, over all trials the average assembly utilization was 86.1% whereas the average delivery utilization was 64.7%. This suggests that the optimal assembly to delivery robot ratio for this task is roughly two to three, although it is unlikely that the marginal utilization of additional robots is strictly linear. This is examined more later.

In our prior work, the average utilization across all robots was roughly 54%[21]. In the trials with the new platform, the overall utilization was 75.4%. This decrease in idle time is desirable, because it minimizes wasted resources.

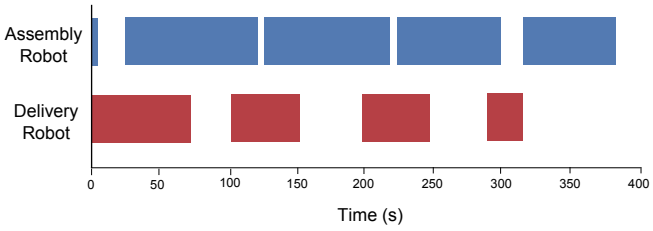


Fig. 6. Robot activity over time in trial 4. Solid blocks of color indicate when a robot was busy with a task, as opposed to idle.

Surprisingly, there is very little correlation between the assembly utilization and delivery utilization (r=0.08). This suggests that the efficiency of one robot is not impacted as much by the length of time it is waiting for another robot to finish a task, but is perhaps influenced by how quickly it is able to finish its own work - and therefore how much it must wait for the next step in the process.

VIII. Three-Dimensional Results

A. Overall Results

Our next task uses the same "log cabin" style, but with six layers instead of two. An example of this structure can be seen in figure Figure 3. This new assembly process requires a total of twelve delivery-assembly iterations per trial. There can also be stability complications, as parts form the foundation for more parts; an early misplacement can cause the entire structure to fall. These trials also increased complexity by using four robots instead of two. Two robots were designated as delivery robots, and the other two as assembly robots.

#	Runtime (M:SS)	Assm. 1 utiliz.	Assm. 2 utiliz.	Del. 1 utiliz.	Del. 2 utiliz.	Succ. (Y/N)
1	9:16	84.8%	90.1%	95.4%	86.3%	Y
2	8:53	88.9%	87.3%	96.6%	96.1%	Y
3	8:46	89.9%	88.7%	93.0%	94.2%	Y
4	10:29	83.3%	86.8%	90.9%	94.4%	Y
5	12:34	87.8%	87.9%	82.0%	92.7%	Y
6	10:33	87.2%	87.9%	96.7%	93.9%	Y
7	8:42	89.7%	87.2%	97.9%	94.0%	Y
8	11:23	93.0%	84.9%	81.5%	93.7%	N
Avg	10:05	88.1%	87.6%	91.8%	91.9%	87.5%

TABLE II
SUMMARY OF FOUR-ROBOT ASSEMBLY TRIALS FOR A TOWER.

Otherwise, all aspects of the assembly process are the same as described in Section VII. The results from the three-dimensional construction are summarized in Table II.

In all of the trials except the last, the structure was completed successfully with no errors. On the last trial, there was a timing issue where both assembly robots attempted to place parts simultaneously and the parts collided. The rest of the assembly process continued as planned, although the two collided pieces had to be re-placed manually to support the remainder of the structure. This may have also led to variability in the metrics for that trial, as both assembly robots experienced difficulty placing these parts accurately due to the other robot's movements. The delivery robots needed to wait for this interaction to finish before delivering more parts, decreasing their utilization and lengthening the time of the overall process.

B. Runtime and Efficiency

Average assembly utilization across both robots was 87.9% and average delivery utilization was much higher at 92.8%. This suggests that the relationship between number of robots and the amount of work is not simply linear, but perhaps more complicated. It would seem that for large-scale assemblies the number of delivery robots should be roughly equal to the number of assembly robots, with perhaps slightly more assembly robots to raise the assembly utilization above 90%.

IX. Experimental Results with Part Unavailability

In order to test our part unavailability algorithm, we used a blueprint of a simple set of stairs. There are three steps, and each step has one, two, or three parts respectively. The top part on each step is a "finished" top part, which we represent with a red cube. All of the cubes underneath top parts are "foundation" parts, represented with a blue cube.

We ran three control trials of the assembly process in which parts did not run out. In these trials, the assembly robot first completes the shortest stair, then the middle stair, and then the tallest stair. We then ran three experimental trials in which after the assembly robot places the first red part, the supply of red parts runs out. The assembly robot adapts

Trial	Type	Runtime (M:SS)	Assembly utilization	Delivery utilization
1	Control	8:44	91.5%	96.5%
2	Control	8:56	88.6%	95.8%
3	Control	8:35	87.7%	94.8%
1	Exp	8:18	87.3%	94.9%
2	Exp	8:55	87.4%	98.0%
3	Exp	9:28	89.2%	97.0%
Avg	Control	8:45	89.2%	95.7%
Avg	Exp	8:54	88.0%	96.6%

TABLE III

SUMMARY OF TWO-ROBOT ASSEMBLY TRIALS OF STAIRS.

and places all of the foundation parts. When the red parts are resupplied the assembly robot completes the structure.

A summary of our results for the two sets of trials can be found in Table III. As can be seen, there was no significant difference in time or utilization between the two trials. This indicates that the assembly robots adequately adapted to the loss of part supply and did not lose efficiency.

X. HANDOFF EXPERIMENTAL RESULTS

Over our recorded trials, there were 114 attempts of direct handoffs of the log cabin parts between robots, 112 of which succeeded (98.2% success rate). Assuming that each handoff is an independent event with this probability of failure, the probability of a 6-layer log cabin construction with no errors in handoffs is 83.8%.

The first failure was due to high network latency, such that the location of the delivery robot was miscommunicated to the assembly robot. The second failure was due a loose grip on the part by the assembly robot; while the assembly robot navigated to the target location, the orientation of the part slipped and was therefore not placed correctly.

XI. CONCLUSION

In this paper we extended our assembly ordering algorithms to account for two real-world scenarios: heterogenous timing constraints and a lack of available parts. We presented a new distributed robotic system that takes advantage of our algorithmic developments for involving physical constraints to order the assembly process. The system reflects many of the ideal characteristics of a robotic assembly platform, including scalability, parallelism, and physical knowledge of the environment in which it operates. We used this system to perform fundamental delivery and assembly tasks with a low rate of failure, and analyzed the utilization and efficiency of the robots involved in the process. We hope to apply these techniques to further assembly tasks of more complex assemblies requiring many more robots.

XII. ACKNOWLEDGEMENTS

This project has been supported in part by The Boeing Company, the U.S. National Science Foundation, Emerging Frontiers in Research and Innovation (EFRI) grant #0735953, and MURI SMARTS grant #N0014-09-1051. We are grateful for this support.

REFERENCES

[1] D. Stein, T. R. Schoen, and D. Rus, "Constraint-aware coordinated construction of generic structures," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011 (submitted).

[2] J. Markoff, "Skilled work, without the worker," *The New York Times*, 18 August 2012. [Online]. Available: http://www.nytimes.com/2012/08/19/business/new-wave-of-adept-robots-is-changing-global-industry.html?pagewanted=all

[3] S. kook Yun, M. Schwager, and D. Rus, "Coordinating construction of truss structures using distributed equal-mass partitioning," in *Proc. of the 14th International Symposium on Robotics Research*, Lucern, Switzerland, August 2009.

[4] S. Yun and D. Rus, "Adaptation to robot failures and shape change in decentralized construction." Institute of Electrical and Electronics Engineers, 2010.

[5] A. Bolger, M. Faulkner, D. Stein, L. White, S. kook Yun, and D. Rus, "Experiments in decentralized robot construction with tool delivery and assembly robots," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[6] S. kook Yun, D. A. Hjelle, H. Lipson, and D. Rus, "Planning the reconfiguration of grounded truss structures with truss climbing robots that carry truss elements," in *Proc. of IEEE/RSJ IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[7] ——, "Planning the reconfiguration of grounded truss structures with truss climbing robots that carry truss elements," in *Proc. of IEEE/RSJ IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[8] S. kook Yun and D. Rus, "Optimal distributed planning for self assembly of modular manipulators," in *Proc. of IEEE/RSJ IEEE International Conference on Intelligent Robots and Systems*, Nice, France, Sep 2008, pp. 1346–1352.

[9] S. kook Yun, "Coordinating construction by a distributed multi-robot system," Ph.D. dissertation, MIT, 2010.

[10] S. kook Yun and D. Rus, "Distributed coverage with mobile robots on a graph Locational optimization and equal-mass partitioning," in *Workshop on the Algorithmic Foundations of Robotics*, 2010.

[11] T. Yasuda, K. Ohkura, and K. Ueda, "A homogeneous mobile robot team that is fault-tolerant," *Advanced Engineering Informatics*, vol. 20, no. 3, pp. 301 – 311, 2006, ¡ce:title¿Design of Complex Adaptive Systems¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474034606000048

[12] H. V. Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters, "Reference architecture for holonic manufacturing systems: Prosa," *Computers in Industry*, vol. 37, no. 3, pp. 255 – 274, 1998.

[13] S. Berman, Á. M. Halász, M. A. Hsieh, and V. Kumar, "Optimized stochastic policies for task allocation in swarms of robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[14] L. Matthey, S. Berman, and V. Kumar, "Stochastic strategies for a swarm robotic assembly system." in *Proc. of IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1953–1958.

[15] M. A. Hsieh and J. Rogoff, "Complexity measures for distributed assembly tasks," in *Proc. of the 2010 Performance Metrics for Intelligent Systems Workshop (PerMIS09)*, Baldimore, MD, USA, Sept 2010.

[16] K. Petersen, R. Nagpal, and J. Werfel, "Termes: An autonomous robotic system for three-dimensional collective construction," in *Proc. of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[17] J. Werfel and R. Nagpal, "Three-Dimensional construction with mobile robots and modular blocks," *Int. J. Rob. Res.*, vol. 27, no. 3-4, pp. 463–479, 2008. [Online]. Available: http://www.eecs.harvard.edu/˜rad/ssr/papers/ijrr08-werfel.pdf

[18] P. White, V. Zykov, J. Bongard, and H. Lipson, "Three dimensional stochastic reconfiguration of modular robots," in *Robotics: Science and Systems*, Cambridge, 2005, pp. 161–168.

[19] Locomotec. (2012, Jan.) Kuka youbot store. youbots. [Online]. Available: http://www.youbot-store.com/category/53-youbots.aspx

[20] R. A. Knepper, S. S. Srinivasa, and M. T. Mason, "Hierarchical planning architectures for mobile manipulation tasks in indoor environments," in *Proc. of International Conference on Robotics and Automation*, May 2010.

[21] D. Stein, March 2012, interview with author of [5].