Errors in Collective Robotic Construction

Jiahe Chen[1⋆], Yifang Liu[1⋆], Adam Pacheck[2⋆], Hadas Kress-Gazit[2], Nils Napp[1], and Kirstin Petersen[1]

[1] School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853, USA, {jc3472, yl892, nnapp, kirstin}@cornell.edu
[2] Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA, {akp84, hadaskg}@cornell.edu

Abstract. We investigate the effect of errors in collective robotic construction (CRC) on both construction time and the probability of correctly completing a specified structure. We ground our investigation in the TERMES distributed construction system, which uses local sensing and stigmergic rules that enable robots to navigate and build 3D structures. We perform an in depth analysis and categorization of action failures in CRC systems. We present an approach to mitigating action failures and preventing errors that prohibit completion of a structure by adding predictive local checks. We show that the predictive local checks can increase the probability of success by orders of magnitude in large structures. This work demonstrates the need to consider both construction time and the effect of errors in collective robotic construction.

1 Introduction

Collective robotic construction (CRC) involves multiple robots collaborating to build structures much larger than themselves [1]. This approach can enable construction by teams of dispensable robots in places dangerous or inaccessible to humans, such as disaster sites or extraterrestrial environments. CRC is a growing research field spanning complex industrial robots cooperating through detailed global planners to simple, locally-aware robots cooperating through a combination of global plans and stigmergic rule sets [2–7]. The latter are of special interest in task settings that provide little pre-existing infrastructure yet demand scalability in deployment, high redundancy, and speed through parallelism. Applications range from building pre-determined structures using traditional materials (e.g., clay bricks and concrete) or custom bricks to ease robot manipulation [8], to functional structures such as access paths built out of amorphous materials [9, 10]. Most of these systems focus purely on additive manufacturing, which makes occasional errors especially problematic since they cannot be undone.

Due to the relatively small scale of robot-built structures shown in the past, errors and error propagation in CRC have been largely overlooked. The majority of prior work has focused on decreasing construction time through improving system efficiency and maximizing parallelism [3–5, 7] while relying on systems where errors are mitigated through engineering effort or simulations that do not incorporate errors. However, in bigger structures and larger collectives, even unlikely errors are bound to occur and it is critical to understand how these affect the overall system. Similar tendencies were shown in the related field of swarm robotics when they grew in numbers from tens to a thousand [11]. Comparatively, errors in CRC often have bigger repercussions because

⋆ The first three authors contributed equally to this work.

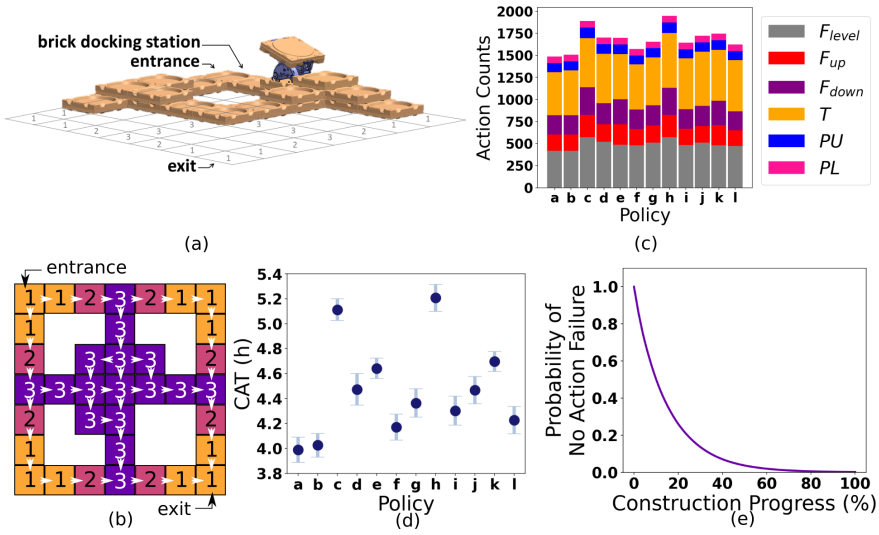

Fig. 1: (a) Sketch of a TERMES robot assembling a structure. b) A blueprint with an example policy overlaid. (c) The composition of actions required to complete (b) given different policies. (d) The cumulative action time (robot hours) required to build (b) given different policies. (e) The probability of completing (b) without having an action failure, using policy 'a'.

robots with limited sensing and motion capabilities are physically and permanently altering their environment. Therefore, when evaluating CRC systems, the probability of successful structure completion must be considered in addition to construction time.

In this work, we focus explicitly on CRC of pre-determined structures, with local stigmergic rules that have a non-zero probability of action failure (Fig. 1). When actions fail, the effect on the collective construction effort varies. While some action failures make no difference beyond the wasted effort, others can lead to a partial structure that is impossible to complete, a situation which we term a *fatal error*. We show that by analyzing which action failures are most likely to cause fatal errors, we can make informed modifications to the system and improve the probability of success. First, we discuss an intuitive way to increase the probability of success by using policies that minimize the number of required sequential actions. We show that this is closely related to finding the policy that results in the lowest construction time, and investigate the difference between a parallel policy that minimizes the number of sequential actions, but may lead to wasted trips without assembly actions, versus a sequential policy where robots are always able to perform assembly actions. The approach of minimizing action sequences, however, still has a fundamental limit based on structure size and robot reliability. We further show that we can improve the probability of success by adding predictive local checks to detect when an action fails, thereby preventing errors from cascading. We show that while adding predicitive local checks can substantially improve reliability, it too has a fundamental limit to its practical application due to an increase in construction time. While preventing fatal errors, structures may never be completed because robots abandon trips when action failures occur.

Action	Success rate ($Pr(a)$)	Execution time
Move forward (F_{level})	0.998	7.6 s
Climb up (F_{up})	0.999	18.96 s
Climb down (F_{down})	0.9999	10.88 s
Turn 90 degrees (TU)	0.986	6.4 s
Pick up a brick (PU)	1	5.6 s
Place a brick (PL)	0.998	21.6 s

Table 1: Action success rate and execution time in our simulated TERMES system. We chose values proportional to those reported for the real TERMES system [8], but decreased the failure probability by 20% to enable more interesting outcomes.

We study these implications in the context of the TERMES system [6] (Fig. 1(a)), an autonomous multi-robot system with distributed control in which robots use only local sensing and follow a stochastic plan to place bricks to build a structure. Although we use the TERMES system as a platform to ground our findings, many of the insights gained translate to other CRC systems, especially the critical importance of considering the system error characteristics when robots with limited sensing and motion constraints manipulate a shared environment.

Contributions: **I** An in depth analysis and categorization of action failures and fatal errors in CRC systems; **II** A general approach to mitigating action failures and preventing fatal errors by using predictive local checks; **III** An evaluation of construction times and success rates for the TERMES system based on published error rates as well as examples of predictive local checks that can increase the probability of success by orders of magnitude, especially in large structures.

2 Terminology

This section briefly summarizes background information and terminology. We focus on CRC systems in which robots with local information move over and assemble predetermined structures, as in [6] and shown in Fig. 1. For clarity, we define the following terms, and include examples of how these apply to the TERMES system:

Blueprint: The desired structure is an N×M grid, where each grid cell, or *location*, is associated with a number representing the number of bricks at that location, and if they serve as a structure entrance or exit (Fig. 1(b)). For simplicity, we assume that the entrance and exit are always located in the upper left and lower right corner, respectively. Note that the robot always picks up a brick at the entrance. In the following, an N×N×1 blueprint is a square structure of height 1 without any holes.
Path: The sequence of locations visited by the robot as it navigates through a structure.
Policy: The blueprint is passed to a compiler which generates one or more *solutions* specifying possible transitions between locations (Fig. 2(a)). The policy combines solutions with *transition probabilities*, such that transition probabilities out of a location always sum to one [7]. Robots travel on top of the structure from the entrance to the exit by following these stochastic policies.
Child and parent locations: The parents of a location are those for which the robot has a non-zero probability of transitioning from; the children are those which the robot has a non-zero probability of transitioning to. Parents and children are specific to policies.

Assembly rules: The assembly rules define valid *assembly locations*, where it is valid to place a brick, based on locally available information. A TERMES robot can place a brick in a location if and only if [6]: 1) The location height is less than the blueprint-specified height; 2) All parents are of height greater than the location, or have reached their blueprint-specified height; 3) All children are of equal height to the location, or their blueprint-specified height differ from the location blueprint-specified height by more than 1. If these are true, the robot executes a sequence of placement actions.
Actions: Robots have a set of actions. In TERMES, these actions correspond to "move forward", either on level bricks or by climbing one brick up or down (F), "turn 90 degrees left or right" (TU), "pick up a brick" (PU), or "place a brick" (PL). The sequence of placement actions is always either $TU{-}F{-}TU{-}TU{-}PL$ or $F{-}TU{-}TU{-}PL$.
Action failure: Physical robots have a non-zero probability of failing to correctly perform an action. The TERMES failure characteristics are reported in [6] and listed in Table 1, where $Pr(a)$ is the probability of action a executing successfully. For simplicity, we assume that failure of locomotion actions means that the robot believes it has performed the action correctly, but physically remains in its current pose. For errors in the placement action (PL), the robot is assumed to have made an error placing that brick and the structure is no longer traversable.
Cumulative Action Time (CAT): The total robot time taken to complete a structure, given by the average action execution time ([6], Table 1). Note that the actual construction time will depend on the number of robots deployed.
Probability of Success (PoS): Probability that a structure will be completed, given the particular failure characteristics of the robots and policy.
Productive and wasted trips: In a productive trip, a robot finds a legal assembly location and correctly places a brick; in a wasted trip the robot does not place a brick.

3 Influence of Policy Choice on System Performance

The number of policies can grow exponentially with the size of the structure, e.g. a $3{\times}3{\times}1$ structure has 5 policies while a $5{\times}5{\times}1$ structure has 1010 policies. Each policy may lead to a different CAT and PoS due to different lengths of action sequences and number of wasted trips. We modified the TERMES simulator from [7] to investigate which policies perform the best given the failure characteristics in Table 1. We pay special attention to two policies: *parallel* and *sequential* (Fig. 2(a)). Parallel policies minimize the longest path robots can take through the structure, and further facilitate the maximum number of simultaneously legal assembly locations. Sequential policies have a single path and thus force robots to visit every assembled location on every trip but eliminate wasted trips. Note that these policies are not necessarily unique, and that not every blueprint has a sequential policy. For example, a square structures with sides of even length and opposing entrance and exit locations lacks a sequential policy.

We found that for larger structures, the benefit of minimizing the path lengths outweighs the risk of wasting trips, making the parallel policies superior to the sequential policies in terms of both PoS and CAT, as shown in Fig. 2(b,e). Fig. 2(c,f) show that the parallel policy also consistently outperforms other randomly selected policies for a $7{\times}7{\times}1$ blueprint in terms of both PoS and CAT. Similarly, in Fig. 2(d) we show that the parallel policy has a higher PoS than 19 randomly selected policies and the sequential

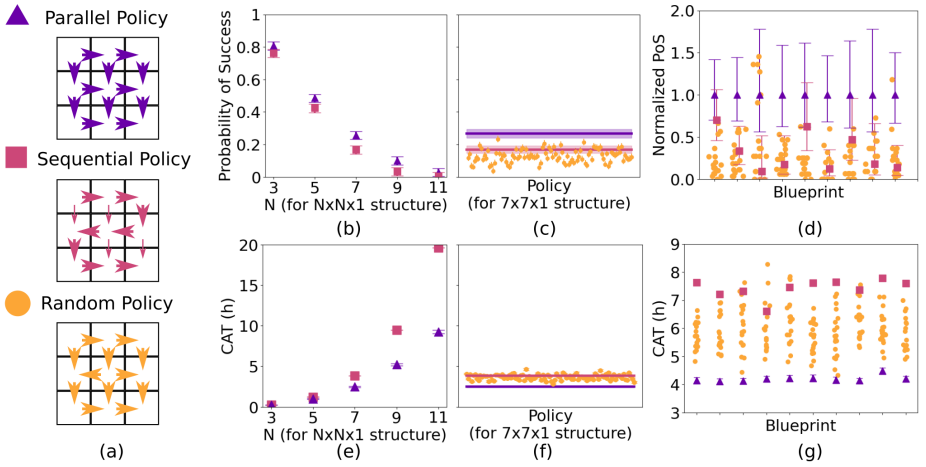

Fig. 2: The PoS and CAT for different policies, based on 1000 and 100 simulations per policy respectively. (a) Example transitions between locations for a parallel, sequential, and random policy. Smaller arrows denote small ($\sim$0) transition probabilities. (b,e) Parallel versus sequential policy for N$\times$N$\times$1 blueprints. (c,f) Parallel, sequential, and 100 random policies for a 7$\times$7$\times$1 blueprint. Purple and pink lines denote the parallel and sequential policies; shaded regions the 95% confidence interval (in (f) the error bars are too small to be seen). (d,g) Parallel, sequential, and 19 randomly selected policies for ten 7$\times$7 blueprints with random heights between 0 and 3.

policy for eight out of ten 7$\times$7 blueprints with random heights. In Fig. 2(g) we show that the CAT of the parallel policy is lower than other policies on all tested blueprints. As parallel policies are (usually) the better option in terms of both CAT and PoS, this will be our focus for the remainder of the paper.

For the structures and policies we tested, the parallel policy has the lowest CAT and usually the highest PoS, but still only allows practical construction of relatively small structures; a 9$\times$9$\times$1 structure, for example, has 10.1% PoS. For comparison, an average American family house consists of 8,000 bricks [7]. To enable such large-scale structures, we need to investigate how errors are caused and methods to mitigate them.

4 Error Analysis

In this section, we investigate how action failures reduce the PoS. It is important to note that the probability of completing a structure without any action failures happening is much smaller than the PoS. The PoS for a 9$\times$9$\times$1 structure is 10.1%, as determined by 1000 simulations. However, on average it takes 2885 actions to complete the same structure and the probability of each of these actions completing successfully is approximately $0.999^{2885} \approx 0$. The reason for this mismatch is that not all action failures prevent a structure from being completed; only fatal errors prevent structure completion. A *fatal error* occurs when a brick is mistakenly placed such that robots are no longer able to physically move through the structure to place additional bricks, or because the combination of the construction state and assembly rules prevent future placements. In

the TERMES system, fatal errors are due to the fact that robots cannot climb more than one brick at a time or fill gaps that are restricted on both sides by other bricks.

To further analyze the impact of action failures and fatal errors on the PoS, we define three categories for the consequences of action failures:

Category I: No brick is placed, either because the robot never finds a valid assembly location or because the robot is capable of detecting the failure and leaves the structure. Both lead to a wasted trip.

Category II: A brick is placed without violating the assembly rules, either by luck (Fig. 3(a)) or the robot detecting a failure and finding a new valid assembly location.

Category III: A brick is placed causing a fatal error, either because the placement itself failed or because one or more prior action failures led to a brick placement that violates the assembly rules. This category can be further divided into 3 cases:

1. Motion constraint violation, because the newly placed brick forms a cliff;
2. Manipulation constraint violation, because the newly placed brick forms an unfillable gap in the structure (Fig. 3(b)), or because the robot attempts to place the brick at a different height from where it is standing (Fig. 3(d));
3. Child locations are assembled before their parents (Figs. 3(c,e,f)); although a robot is physically capable of resolving the error, doing so would violate the assembly rules.

Among these categories, our primary interest is in category III, as the consequences of failures that lead to this category directly impact the PoS. It is important to note that an action failure may occur several steps before the fatal error occurs. For example, in Fig. 3(f) the robot attempts to move forward, but fails. It then takes another step forward, turns, and places a brick at what it thinks is the beginning of a row, but is actually the middle of a row. This results in a fatal error, because a robot can never legally place a brick at the far right location of the middle row.

Using our simulator, we recorded which action failures most commonly lead to fatal errors given different policies (Table 2 rows 2-4). We distinguish between action failures that occur during navigation and the brick placement action sequence. Failures to turn (TU_e) (Table 2 columns 4 and 5) result in more than 70% of the fatal errors across different policies. The number of placement failures (PL_e) is stable since it only depends on the number of attempted brick placements. The rest of the fatal errors are

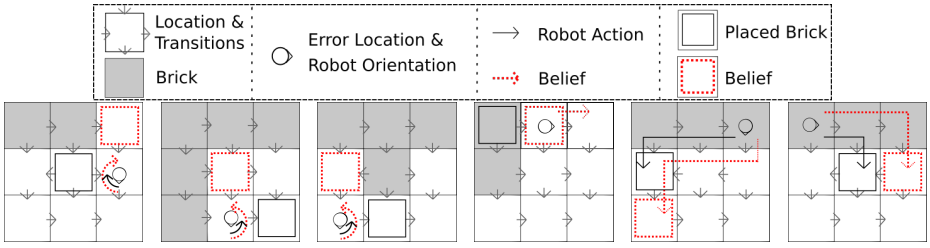

Fig. 3: Examples of action failures. (a-c) show failure to turn (TU_e); (d-f) show failure to move forward (F_e). (a) does not lead to a fatal error; (b-f) lead to fatal errors.

caused by failures to move forward (F_e), either during navigation or placement. Different policies also have different error distributions. For the parallel policy, the percentage of fatal errors caused by F_e failures during navigation is 0.78%, as opposed to 17.3% for the sequential policy. This is because the sequential policy enforces much longer paths, increasing the probability of F_e failures.

Based on analyzing the action failures that lead to fatal errors through the examples shown in Fig. 3 and Table 2, we can determine if and how a robot can detect and mitigate the effects of action failures. If a robot can predict what the surroundings should look like after an action, then it can recognize action failures that cause a mismatch between the expected surroundings and what is physically sensed, and react accordingly. In the next section, we discuss how to mitigate errors by adding such local predictive checks based on the failures shown in Fig. 3.

5 Mitigating Errors through Predictive Local Checks

The ability to sense and reason about legal assembly locations given knowledge of only the local environment is critical in stigmergy coordinated CRC. Here, we suggest several additional local checks a robot can perform to detect if an action failure has occurred, either immediately or after a sequence of actions. The goal of these checks is to convert action failures from category III to category I before they lead to a fatal error. Theoretically, turning category III failures into category II failures would be a better choice, but that requires significant additional effort in relocalizing, estimating the true assembly state, and probabilistic reasoning, so we leave this for future work. In this section, we first discuss predictive local checks to improve the PoS, then discuss how additional wasted trips caused by detected action failures impact the CAT.

5.1 Predictive Local Checks

We propose predictive local checks that compare what the robot expects to sense following a successful action to what it actually senses. This allows a robot to detect some failures to move forward and turn. Here, we assume that the robot can navigate off the structure once a failure is detected without changing the current construction state even if its localization is wrong. The difficulties in designing predictive local checks are that sometimes the local view of the construction state is the same before and after an action, and that the exact construction state in non-sequential policies is not known as actions can be completed in many different orders. As shown in Table 2, the majority of fatal errors stemming from failures to turn occur during brick placement, which means these must be detected immediately. Conversely, we do not need to immediately detect fatal errors stemming from failures to move forward as the majority of such failures happen during navigation, leaving more steps before a brick is placed. We do not attempt to avoid errors stemming from placement failures (PL_e), as these are only associated with hardware reliability and cannot be mitigated by better execution plans. Considering the effort and cost of re-designing the hardware, we focus on local checks that can be accomplished with minimal hardware changes or entirely in software. Specifically, we propose the following two predictive local checks to avoid fatal errors:

Front checking: This check is implemented entirely in software. The TERMES robots have two sensors that permit them to reason about the height of the current and

		solution	$TU_{e,n}$	$TU_{e,p}$	$F_{e,n}$	$F_{e,p}$	PL_e	unsuccessful builds	successful builds
2	TERMES baseline	parallel	0	768	7	62	57	894	166
3	TERMES baseline	sequential	2	837	198	53	56	1146	104
4	TERMES baseline	random	106	770	130	65	54	1125	105
5	w/ predictive checks	parallel	0	392	3	2	53	450	304
6	w/ predictive checks	sequential	0	224	165	10	55	454	326
7	w/ predictive checks	random	0	284	81	13	50	428	322

Table 2: We had a robot repeatedly construct a 7×9×1 blueprint until it successfully placed 30,000 bricks. This table shows the number of action failures that directly lead to fatal errors, without (rows 2-4) and with (rows 5-7) predictive local checks. Subscript e indicates an action failure, and subscript p and n means that the failure occurred during brick placement or navigation, respectively.

neighboring location: a tilt-sensor that indicates when the robot is climbing up or down and a downward-facing sensor mounted on the claw that detects the relative height of a neighboring location. Using these sensors, we can ensure that the robot never attempts to place a brick at a height different from its own location. This means that the types of errors shown in Fig. 3(d) are eliminated. Similarly, the robot can detect the action failures shown in Fig. 3(e,f). In Fig. 3(f), when the robot believes it is facing right at the top right corner, the height of the front location is outside of the structure and should be lower than the robot's current height. If not, the robot can determine that one or more actions failures occurred. Similarly, in Fig. 3(e), when the robot is at the top right corner facing downwards, it can detect that going forward should result in a climb down, when it doesn't, the robot knows an error occurred.

Side checking: By adding three distance sensors to the left, right, and rear of the robot, similar to the front distance sensor that already exists, the robot can detect discrepancies associated with failed turns, such as the ones shown in Fig. 3(b).

By incorporating front and side checking, we are able to reduce the number of fatal errors and increase the PoS. As shown in Table 2 rows 5-7, the number of F_e and TU_e decreases. Most significantly, the number of turn failures during placement decreased by over 45%. Additionally, no fatal errors resulted from turn failures during navigation in all tested policies. The addition of predictive local checks increases the PoS for all policies (Fig. 4). The PoS for parallel and sequential policies on N×N×1 structures improved by an average of 2.6($\pm$1.7) times and 9.7($\pm$12.2) times, respectively (Fig. 4(a)). The PoS of random policies becomes much closer to that of parallel polices, with some random policies even outperforming the parallel policy. The PoS of policies for a 7×7× [1, 3] random structure as shown in Fig. 4(c) improved by 100.5($\pm$95.3) times. This occurs because most action failures that lead to fatal errors in the parallel policy arise from turn failures like the one in Fig. 3(c), which the robot cannot detect right away, as all the surrounding locations have the same relative height, while all other turn failures that occur more often in random and sequential policies can be detected immediately.

Notice that not all action failures can be detected by the robot, even with the virtually updated hardware. For instance, the robot cannot immediately detect a failure to move forward unless the front location has a different height. Even in this case, the robot may still detect that a failure has occurred later on. The robot also cannot immediately detect a failure to turn if all the surroundings have the same relative height.

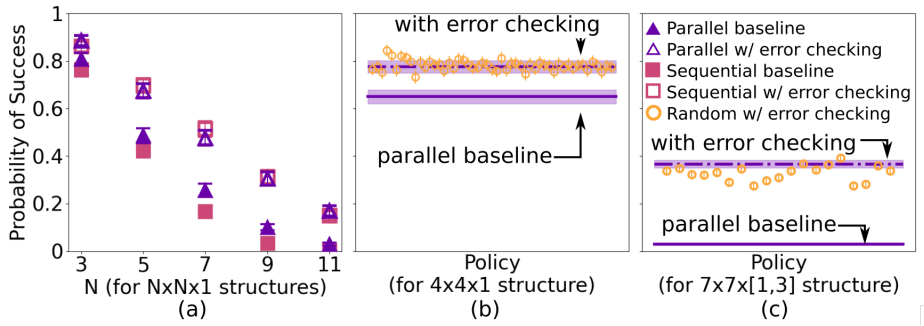

Fig. 4: Adding predictive local checks improves the PoS. Each policy was simulated 1000 times (except for the $11\times11\times1$ blueprint, which was simulated 2000 times) (a) The PoS for both parallel and sequential policies increases with the additional checks. (b) The PoS for all policies is higher than the original PoS of the parallel policy (solid purple line, shaded region is error), and similar to the parallel policy with additional checks (dashed purple line) for a $4\times4\times1$ blueprint. (c) Additional checks also improve the PoS of a blueprint with randomly varying heights ($7\times7\times[1,3]$).

5.2 Impact of Predictive Local Checks on Construction Time

While predictive local checks significantly improve the PoS, they also impair the CAT by introducing additional wasted trips. To study this impact, we developed an analytical expression for the parallel policy CAT. This method not only allows us to quickly reason about the number of robots to deploy for a given structure; it also allows us to study how the CAT scales with the structure size given predictive local checks, enabling us to consider structures that are too large to effectively simulate.

To introduce the model, we define a snapshot of the structure in time, as it is being built, as a *construction state*. Given a construction state, robots navigate through the structure until they reach a location where a brick still needs to be placed; there can be multiple such locations per construction state. If a robot reaches a location that complies with the assembly rules (a *reachable legal assembly location*), it places a brick and leaves the structure. If a robot reaches a location that violates any assembly rules (a *reachable but non-legal assembly location*), it leaves the structure. The probability of following a particular path in the structure is the product of the transition probabilities between the corresponding locations. We define the duration of a path, excluding the time to place a brick, as *trip time*. The time to place a brick after the arrival is counted separately, as the number of placements is fixed. We do not include the time it takes for the robot to return to the entrance after exiting the structure into account. For a location that can be reached through multiple valid paths, the weighted average trip time is used.

To approximate the CAT, we split the construction process into *construction cycles*. In each cycle, the robot has to visit all reachable legal assembly locations at least once, and a brick will be placed at each location (Fig. 5(b)). Modeling the construction process as a sequence of construction cycles leads to an over-approximation, since in reality once some of the locations have been visited and assembled, new locations can become reachable and "legal", but visiting them still counts as wasted trips in our calculations.

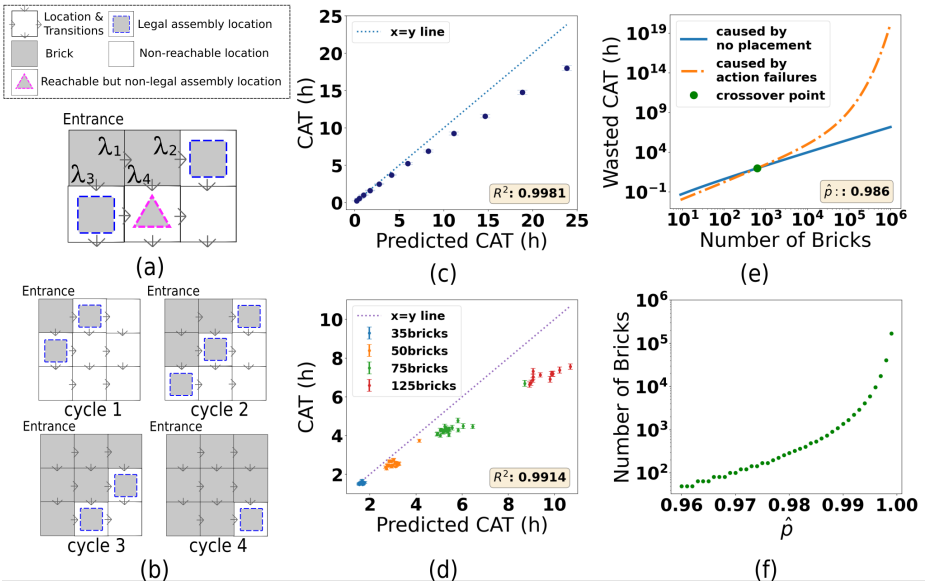

Fig. 5: (a) A possible construction state where λ is the transition probability. (b) Construction cycles for a 3×3×1 blueprint, as assumed in the general model proposed in Eq. 1. (c) Simulated versus estimated CAT obtained from the simplified model in Eq. 2 of N×N×1 ($3 \leq \mathrm{N} \leq 14$) blueprints simulated 100 times. (d) Simulated vs estimated CAT obtained from the general model in Eq. 1 of N×N ($5 \leq \mathrm{N} \leq 7$) blueprints with random heights between 0 and 3 simulated 100 times. (e) Wasted CAT caused by action failures and no placement at different structure size. $\hat{p}$ is set to the action success rate of turning 90 degrees (Table 1). (f) Crossover point at different $\hat{p}$.

We can now produce a closed form expression for a CAT over-approximation. For each construction cycle d, we denote the weighted average trip time of all reachable legal assembly locations and all reachable but non-legal assembly locations as $t_{I,d}$ and $t_{J,d}$ respectively; we denote the probability of visiting any reachable legal assembly location as $p_{I,d}$. We consider two random variables: X_d as the number of trips needed to visit all reachable legal assembly locations at least once given that only reachable legal assembly locations will be visited, and Y_d as the number of trips needed until a reachable legal assembly location is visited. Since visiting a location does not affect the probability of visiting any other location, X_d and Y_d are independent. The expected number of trips to visit all reachable legal assembly locations at least once is then $\mathrm{E}[X_d]\,\mathrm{E}[Y_d]$. The expectation of Y_d is $1/p_{I,d}$. Since only one out of the $\mathrm{E}[Y_d]$ number of trips goes to a reachable legal assembly location, the trip time is $(\mathrm{E}[Y_d]-1)t_{J,d}+t_{I,d}$. Computation of $\mathrm{E}[X_d]$ can be cast as a general *coupon collector's problem*. Each location can be considered as a coupon with certain probability to collect it and $\mathrm{E}[X_d]$ is the expected number of trials needed to collect all of them. A closed form solution has been proposed in [12](Eq. 14b) and is used in our model. Then for a construction process that has D construction cycles, the estimated expected CAT $\mathrm{E}[T]$ is given by the general model:

$$\mathrm{E}[T] = T_{PL} + \sum_{d=1}^{D} \mathrm{E}[X_d]\Big((\frac{1}{p_{I,d}} - 1)t_{J,d} + t_{I,d}\Big), \tag{1}$$

where $T_{PL} = 48.4N_B$ is the total brick placement time in seconds and N_B is the total number of bricks that need to be placed in the structure. To place a brick the robot will execute 1 move, 1 placement, and at most 3 turns which amounts to a maximum of 48.4s (Table 1). Fig. 5(d) compares the expected CAT obtained from the general model (Eq. 1) and simulation results and shows that although the model is an over-approximation of the expected CAT, it has a linear relationship with the simulation result and can be used to predict CAT growth as a function of the structure size.

For flat, square structures with the parallel policy, each construction cycle contains only reachable legal assembly locations, and each location has equal probability of being visited (Fig. 5(b)), enabling a direct mapping to a closed-form solution from [12]: for n locations, the expected number of trips until all locations have been visited at least once is nH_n, where $H_n = \sum_{k=1}^{n} \frac{1}{k}$ is the n^{th} harmonic number. For an N×N×1 structure, there are $2(N-1)$ construction cycles and the number of reachable legal assembly locations in each cycle is $\{2, 3, ..., N-1, N, N-1, ..., 2, 1\}$. For each construction cycle d, a trip contains 1 pickup, d moves and on average $2 + d/4$ turns. Thus, for an N×N×1 structure with the parallel policy, the CAT over-approximation $\mathrm{E}[T_{\mathrm{N\times N\times 1}}]$ is:

$$\begin{aligned} \mathrm{E}[T_{\mathrm{N\times N\times 1}}] &= T_{PL} + \sum_{d=1}^{2(N-1)} \mathrm{E}[X_d]\; t_{I,d}, \\ \text{with} \quad \mathrm{E}[X_d] &= \begin{cases} (d+1)H_{d+1} & \text{for} \quad 1 \le d \le N-1 \\ (2N-1-d)H_{2N-1-d} & \text{for} \quad N \le d \le 2(N-1) \end{cases}, \\ t_{I,d} &= \tau_F\, d + \tau_{TU}(2 + d/4) + \tau_{PU}, \end{aligned} \tag{2}$$

where τ_F, τ_{TU} and τ_{PU} are the execution time of move, turn, and pickup, respectively (Table. 1). Fig. 5(c) shows that the simplified model still maintains a linear relationship with the simulation results. We will use this simplified model to study the scaling behavior of the CAT due to its computation efficiency. We can also express the expected time of trips during which a robot actually places a brick (*productive CAT*) for an N×N×1 structure with a parallel policy in Eq. 3.

$$\mathrm{E}[T^{\text{productive}}_{\mathrm{N\times N\times 1}}] = T_{PL} + \sum_{d=1}^{2(N-1)} n_{I,d}\; t_{I,d}, \tag{3}$$

where $n_{I,d}$ is the number of reachable legal assembly locations in construction cycle d; $t_{I,d}$ is the same as in Eq. 2. The time over the productive CAT is *wasted CAT*.

To examine the effect of predictive local checks, we assume an ideal scenario where a robot can detect all action failures and then leave the structure. For simplicity, we assume that every action has the same average success rate $\hat{p}$. For each construction cycle d, we define a random variable Z_d as the number of trials needed until a "successful" trip, i.e. a trip without failures, occurs. Z_d and the two random variables X_d and Y_d defined in the general model (Eq. 1) are independent. Therefore, the expected number of trips needed to visit all reachable legal assembly locations at least once successfully

is $\mathrm{E}[X_d]\,\mathrm{E}[Y_d]\,\mathrm{E}[Z_d]$. We denote the weighted average number of actions in the trips to all reachable legal assembly locations as $a_{I,d} = 3 + 5d/4$. Then $\mathrm{E}[Z_d] = 1/\hat{p}^{a_{I,d}}$. To further simplify the model, we assume that action failures do not occur during brick placement since that time only amounts to a small portion of the total CAT. Under these assumptions, the CAT over-approximation with predictive local checks $\mathrm{E}[T^{\text{checks}}_{\mathrm{N}\times\mathrm{N}\times 1}]$:

$$\mathrm{E}[T^{\text{checks}}_{\mathrm{N}\times\mathrm{N}\times 1}] = T_{PL} + \sum_{d=1}^{2(N-1)} \mathrm{E}[X_d]\ \frac{1}{\hat{p}^{a_{I,d}}}\ t_{I,d}, \tag{4}$$

Eq. 4 tells us that the CAT can be wasted due to both action failures, and the situation where there is "no placement" because the robot does not find a valid assembly location. The wasted CAT caused by action failures is $\mathrm{E}[T^{\text{checks}}_{\mathrm{N}\times\mathrm{N}\times 1}] - \mathrm{E}[T_{\mathrm{N}\times\mathrm{N}\times 1}]$ and the wasted CAT caused by no placement is $\mathrm{E}[T_{\mathrm{N}\times\mathrm{N}\times 1}] - \mathrm{E}[T^{\text{productive}}_{\mathrm{N}\times\mathrm{N}\times 1}]$. Fig. 5(e) compares the wasted CAT associated with each, and shows that as the structure size increases, the effect of action failures grows and ultimately surpasses the effect of trips without placement. We define the structure size at which the action failures and the trips without placements lead to the same wasted CAT as the *crossover point*. Beyond this point, action failures become the dominant cause of wasted CAT and the scaling behavior of the CAT becomes undesirable. Therefore the crossover point can also be interpreted as the structure size limit under which predictive local checks are a good strategy. Fig. 5(f) shows the crossover point at different $\hat{p}$ and supports the intuitive notion that improving the action success rate can significantly shift the crossover point higher, making the system more capable of building larger structures within reasonable time span.

6 Conclusion

In this work, we investigated the effect of errors in CRC. Grounding our work in the TERMES system, we have shown that the parallel policy, compared to a sequential or random policy, is in general the policy that will yield the highest probability of success and lowest cumulative action time. Additionally, we proposed a categorization of mistakes into action failures and fatal errors, showing how certain action failures are more likely to yield fatal errors. Using insights gained from the investigation of errors, we proposed predictive local checks requiring software modifications and minimal hardware changes. We showed how these predictive local checks substantially increase the probability of success, in some cases by an order of magnitude. Adding these checks also reduces the differences in success probability between the various policies, so that the worst ones are the most improved. In addition, we showed that these local predictive checks can result in an increase in cumulative action time.

This work has many potential future directions. Here, we considered action failures that do not change the robot state; next, it would be interesting to investigate more complex classes of failures or sensing errors. We showed that when constructing large structures, local predictive checks increase the probability of success, while increasing the cumulative action time. It would also be interesting to develop localization methods that allow the robots to recover from failures instead of simply detecting them.

Acknowledgments

This project was funded by the Packard Fellowship for Science and Engineering, GETTYLABS, and the National Science Foundation (NSF) Grant #1846340 and #2042411.

References

[1] K. H. Petersen, N. Napp, R. Stuart-Smith, D. Rus, and M. Kovac, "A review of collective robotic construction," *Science Robotics*, vol. 4, no. 28, 2019.

[2] J. Solly, N. Frueh, S. Saffarian, M. Prado, L. Vasey, B. Felbrich, D. Reist, J. Knippers, and A. Menges, "Icd/itke research pavilion 2016/2017: integrative design of a composite lattice cantilever," in *Proceedings of IASS Annual Symposia*. International Association for Shell and Spatial Structures (IASS), 2018, pp. 1–8.

[3] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The flight assembled architecture installation: Cooperative construction with flying machines," *IEEE Control Systems Magazine*, vol. 34, no. 4, pp. 46–64, 2014.

[4] I. O'hara, J. Paulos, J. Davey, N. Eckenstein, N. Doshi, T. Tosun, J. Greco, J. Seo, M. Turpin, V. Kumar *et al.*, "Self-assembly of a swarm of autonomous boats into floating structures," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1234–1240.

[5] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction of cubic structures with quadrotor teams," *Proc. Robotics: Science & Systems VII*, 2011.

[6] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[7] Y. Deng, Y. Hua, N. Napp, and K. Petersen, "A Compiler for Scalable Construction by the TERMES Robot Collective," *Robotics and Autonomous Systems*, vol. 121, 2019.

[8] K. Petersen, R. Nagpal, and J. Werfel, "TERMES: An autonomous robotic system for three-dimensional collective construction," in *Robotics: Science and Systems*, vol. 7, 2012, pp. 257–264.

[9] N. Napp and R. Nagpal, "Distributed amorphous ramp construction in unstructured environments," *Robotica*, vol. 32, no. 2, pp. 279–290, 2014.

[10] T. Soleymani, V. Trianni, M. Bonani, F. Mondada, and M. Dorigo, "Bio-inspired construction with mobile robots and compliant pockets," in *Robotics and Autonomous Systems*, 2015.

[11] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[12] P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229, 1992.