Designing Robot Teams for Distributed Construction, Repair, and Maintenance

TODD WAREHAM, Memorial University of Newfoundland, Canada

Designing teams of autonomous robots that can create target structures or repair damage to those structures on either a one-off or ongoing basis is an important problem in distributed robotics. However, it is not known if a team design algorithm for any of these tasks can both have low runtime and produce teams that will always perform their specified tasks quickly and correctly. In this article, we give the first computational and parameterized complexity analyses of several robot team design problems associated with creating, repairing, and maintaining target structures in given environments. Our goals are to establish whether efficient design algorithms exist that operate reliably on all possible inputs and, if not, under which restrictions such algorithms are and are not possible. We prove that all of our design problems are not efficiently solvable in general for heterogeneous robot teams and remain so under a number of plausible restrictions on robot controllers, environments, and target structures. We also give the first restrictions relative to which some of these problems may be efficiently solvable and discuss how theoretical results like those derived here can be combined with physical experiments to derive the best possible algorithms for real-world robot team design.

CCS Concepts: • **Computing methodologies** → **Multi-agent systems**; **Multi-agent planning**; • **Theory of computation** → **Design and analysis of algorithms**; **Parameterized complexity and exact algorithms**;

Additional Key Words and Phrases: Swarm robotics, construction, computational complexity

ACM Reference format:
Todd Wareham. 2019. Designing Robot Teams for Distributed Construction, Repair, and Maintenance. *ACM Trans. Auton. Adapt. Syst.* 14, 1, Article 2 (July 2019), 29 pages.
https://doi.org/10.1145/3337797

1 INTRODUCTION

The design of robots to aid in the building, repair, and maintenance of real-world structures is a problem of increasing importance in robotics [4, 31, 33]. Of particular interest is the design of teams of robots that can perform these tasks autonomously without human intervention or guidance [3, 19]. This is necessary in situations in which either humans are supervising large numbers of robots and must direct subgroups of robots to perform specified tasks on their own [26] or human supervision is impossible, e.g., creation of structures in deep space or on other planets [37].

Much research on the design of teams of autonomous robots for distributed construction, repair, and maintenance has been done over the last 30 years (with the primary focus to date being on construction) [3, 19]. This work falls primarily into two streams:

This work was supported by NSERC 0001 http://www.nserc-crsng.gc.ca/index_eng.asp Discovery Grant 228104-2015 0002.
Authors' address: T. Wareham, Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada; email: harold@mun.ca.

1556-4665/2019/07-ART2 $15.00
https://doi.org/10.1145/3337797

(1) Understanding existing biological construction systems, e.g., structures built by social insects such as ants, bees, and termites [7]. This involves developing abstract models of these systems whose characteristics are investigated by mathematical analysis and/or computer simulations, e.g., models of 3D nest construction in which robots use a simple stochastic rule to add building blocks to an initial seed-structure [38].
(2) Creating robot teams for building specified structures [9]. This involves developing algorithms and onboard data structures (often inspired by biological construction systems) for the robots in the team such that the team can build the specified structure, e.g., termite-inspired construction in which all robots have the same basic controller acting in concert with an onboard pre-computed structplan, which specifies how individual robots move and place blocks within the building site [50].

These approaches have been very successful in uncovering basic principles underlying biological construction systems, e.g., stigmergy and templates [39], as well as the characteristics of particular controller algorithm/data structure combinations. However, this has not yet resulted in a set of techniques with the efficiency and reliability characteristic of engineering [5, 9, 17]. If the ultimate goal is to automatically and efficiently (in terms of low design-time) create robot teams, which perform construction, repair, and maintenance tasks both reliably (i.e., always perform the task correctly) and efficiently (i.e., always have lowtask performance time), an approach employing powerful techniques from both algorithm design and abstract problem analysis should be useful.

Such an approach can be implemented by answering the following two questions relative to a given construction-related task A and a set C of desired structure/robot team characteristics:

(1) Can a robot team be efficiently designed relative to arbitrary A and C?
(2) If not, under what restrictions on A and C is efficient team design possible?

Answering the first question establishes whether or not efficient design is possible in general and is a sensible prelude to justifying the use of heuristic design methods. Answering the second question yields a systematic overview of those combinations of restrictions for which efficient design techniques (which need not be biologically based) do and do not exist. Such an overview would be an invaluable aid in both deriving the best possible solutions for given construction instances (by allowing lookup of the most appropriate design techniques relative to characteristics of those instances) and productively directing research on deriving new design techniques (by highlighting those situations for which such techniques exist).

These two questions are best answered using computational complexity analysis [13, 18]. Such analyses essentially determine whether or not there is an efficient algorithm for a given problem, i.e., whether that problem is tractable or intractable. So-called classical complexity analysis [18] establishes whether a problem can be solved efficiently in general, i.e., for all inputs. If this is not the case, parameterized complexity analysis [13] establishes relative to which sets of restrictions the problem can and cannot be solved efficiently. In order to have the greatest possible applicability, both such analyses are typically performed relative to simplified versions of problems that are special cases of the actual problems of interest, where a problem Π is a special case of a more general problem Π' if any algorithm for Π' can be used to solve Π. This is of use because intractability results for such special-case problems then imply intractability for all more general problems (see Section 5 for more details).

Parameterized complexity analyses are particularly important because the results of such an analysis for a problem can be used to derive an intractability map [44], which corresponds to the desired systematic overview described above. An example of how this derivation is done is given in Figure 1 for a hypothetical problem Π with restriction-set $\{A, B, C, D\}$. Part (a) of this figure

	R1	R2	R3
A	X	–	√
B	–	X	√
C	X	X	–
D	X	–	–

(a)

⇒

	–	C	D	C, D
–	NPh	X	X	X
A	X	X	X	X^{R1}
B	X	X^{R2}	???	???
A, B	$\surd^{R3}$	√	√	√

(b)

Fig. 1. Parameterized complexity analysis and intractability maps. (a) A set of parameterized intractability (R1, R2) and tractability (R3) results for a problem Π under various restrictions (A, B, C, D). (b) An intractability map derived from the results in (a). See main text for explanation.

describes a set of parameterized intractability (R1, R2) and tractability (R3) results for Π, where each column in this table denotes the result when Π is restricted by the set of restrictions denoted by X's (in the case of intractability results) or √'s (in the case of tractability results). Part (b) gives the intractability map associated with this set of results. Each cell in this map denotes the tractability status of Π when restricted by the union of the sets of restrictions labeling that cell's column and row. The "raw" results from the table in part (a) are denoted by superscripted entries (X^{R1}, X^{R2}, $\surd^{R3}$) and all other X and √ results in the map follow from the observations that (1) intractability relative to a restriction-set X also holds relative to any subset X' of X and (2) tractability relative to a set X of restrictions also holds relative to any superset X' of X (see Section 4 for details). The remaining ???-entries correspond to restriction-sets whose parameterized complexity is not specified or implied in the given results. As there are ???-entries in the map in part (b), this map is thus a partial intractability map.

In this article, we will give the results of the first computational and parameterized complexity analyses of the following problems associated with designing teams of autonomous robots for construction, repair, and maintenance tasks:

— *Team Design for Construction*: Is it possible to efficiently design a robot team that produces a given target structure in a given environment reliably and efficiently?
— *Team Design for Repair*: Is it possible to efficiently design a robot team that repairs a specific type of damage to a given target structure in a given environment reliably and efficiently?
— *Team Design for Maintenance*: Is it possible to efficiently design a robot team that repairs any type of damage to a given target structure in a given environment reliably and efficiently on an ongoing basis?

Our analysis of the first of these problems complements previously-published analyses of robot team design for construction in a given environment relative to robot controller design from scratch [47] and robot team/environment co-design for construction relative to robot controller design by selection from a given library [41]. To ensure the maximum possible applicability for our results, our analyses are done in terms of a simple type of design (design by selecting robots for a team from a provided library of robots [41]), a simple finite-state robot controller model defined in Ref. [47] that moves in a non-continuous manner in a grid-based environment, a simple model of structures as 2D patterns of grid-squares in such an environment, and a simple model of structural damage in which any combination of squares comprising a 2D target-structure can be deleted. Relative to these conceptions, we show that none of the three problems above can be solved efficiently with respect to heterogeneous robot teams and two-dimensional structures either in general or relative to a number of (often simultaneous) restrictions on controller architecture, environment, and target structure. We also give the first known restrictions under which some of

these problems may be efficiently solvable and discuss how theoretical solvability and unsolvability results like those derived in this article can be combined with physical robot experiments to derive the best possible algorithms for real-world robot team design.

1.1 Previous Work

There has been extensive work on developing robots for use in the real-world construction industry over the last 3 decades [4, 31, 33]. To date, almost all of this work has involved single robots that are either tele-operated, supervised, or semi-supervised by human beings; this is attributed to the conservatism of the construction industry as well as justifiable concerns over the safe interaction of robots with human beings in the typically unstructured environment of a construction site [3, 33]. Research on teams of autonomous robots in construction has therefore been almost exclusively an academic endeavor, split into the biological modeling and bio-inspired algorithm development streams described in the previous section. Algorithms proposed to date focus mostly on designing homogeneous robot teams in which individual robots are guided by stochastic behavior rules [8, 20, 29, 34, 38, 43, 49–51, 53] (see also Refs [3], [5], [9], and [19], and other references). Such rules are seen as essential to allowing robots to both mitigate problems associated with uncertain sensing and motion, and operate in a fair and deadlock-free manner. That being said, it is not known if any design algorithm can both be fast and produce teams that are guaranteed to correctly and quickly perform their construction-related tasks.

Various work has been done on the computational complexity of both verifying if a given multi-entity system can perform a task and designing such systems for tasks. The systems so treated include groups of agents [14, 36, 52], robots [12, 22], game-pieces [15], and tiles [2]. Much of this work, e.g., Refs [2], [12], [15], and [22], assumes that the entities being moved cannot sense, plan, or move autonomously. In the work where entities do have these abilities, e.g., Refs [14], [36], and [52], the formalizations of control mechanisms and environments are very general and powerful (e.g., arbitrary Turing machines or Boolean propositional formulae), rendering both the intractability of these problems unsurprising and the derived results unenlightening with respect to possible restrictions that could yield tractability.

Five complexity-theoretic papers to date incorporate both autonomous robots and a suitably simple and explicit model of robot architecture and environment [41, 45–48]. Of these, only two consider construction-related tasks, with Ref. [48] considering construction relative to robot team design in a given environment where robot teams are designed from scratch and Ref. [41] considering construction relative to robot team/environment co-design where robot teams are designed by selection from a provided library. Both of these analyses consider only the initial creation of structures, and do not address repairing subsequent damage to those structures on either a one-off or ongoing basis. Hence, no computational complexity analysis done to date has investigated construction (let alone (one-off) repair or (ongoing) maintenance) tasks in given environments relative to the library-selection conception of robot team design.

1.2 Organization of Article

This article is organized as follows. In Section 2, we describe our robot controller, environment, and target structure models and use these to formalize the robot team design problems for construction, repair, and maintenance that we will analyze. Section 3 demonstrates the general intractability of these problems and Section 4 identifies which of a basic set of restrictions do and do not make these problems efficiently solvable. When they are not given in the main text, proofs of stated results are given in the appendix. The applicability of our results to more complex and realistic controllers, environments, and target structures as well as the implications of these results for both the simple

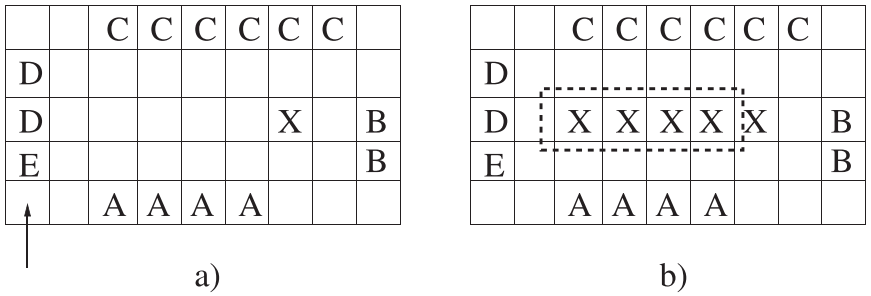

Fig. 2. Environments and structures. (a) A 9×5 environment based on $E_T = \{e_A, e_B, e_C, e_D, e_E, e_X, e_{blank}\}$. All squares not otherwise specified have type e_{blank}. The arrow indicates the center of square $E_{1,1}$. (b) The environment in (a) with the addition of a structure X located at $p_X = E_{3,3}$ (structure enclosed in dashed box).

robot systems analyzed here and real-world robotics are discussed in Sections 5 and 6. Finally, our conclusions and directions for future work are given in Section 7.

2 FORMALIZING STRUCTURE MANIPULATION BY ROBOT TEAMS

In this section, we first formalize the basic entities in our model of structure manipulation by robot teams—namely, environments (Section 2.1), target structures and structural damage (Section 2.2), individual robots (Section 2.3), and robot teams (Section 2.4). Many of these formalizations were originally developed in Ref. [47]. We then formalize the computational problems encoding robot team design for construction, repair, and maintenance in given environments that we will analyze in the remainder of the article (Section 2.5). For each entity and problem, we also discuss how our formalizations of that entity and problem compare with formalizations proposed in the literature and/or used in practice.

2.1 Environments

Our robots exist within a finite square-based environment E in which basic compass movement is possible between adjacent squares, i.e., north, south, east, and west, and each square is either a freespace (which a robot can occupy or travel through) or an obstacle. Let $E_{i,j}$ denote the square that is in the ith column and jth row of E such that $E_{1,1}$ is the square in the southwesternmost corner of E. Each freespace and obstacle square has an associated type that can be sensed by a robot, e.g., grass, gravel, wall; let this set of square-types be denoted by E_T. An example environment is shown in part (a) of Figure 2.

Unlike the continuous environments studied in much of the literature in which entities can occupy any position, our environments are discrete and require that entities be positioned at integer-valued grid-locations. Moreover, our environments are two-dimensional. That being the case, our environments are a special case of 3D grid-based environments populated by 2D squares and 3D blocks, where each square and block has a particular sensed type, squares can be freespaces or obstacles, all blocks are obstacles, and blocks can be piled on top of each other.

2.2 Target Structures and Structural Damage

A structure X in an environment E is a two-dimensional north-oriented pattern of squares in an $m \times n$ grid whose location in E is specified relative to the lower left corner of the grid. All squares in a structure have type e_X, which may be specified as a freespace or an obstacle. An example structure is shown in part (b) of Figure 2.

A damage d_X to a structure X is a non-empty subset of the squares in X. When X has damage d_X, all squares specified in d_X have special type e_{damage}. Let D_X be the set of all possible damages to X. For example, given structure X shown in part (b) of Figure 2 based on squares $E_{3,3}$, $E_{3,4}$, $E_{3,5}$, and $E_{3,6}$, three possible damages to X are $\{E_{3,3}\}$, $\{E_{3,4}, E_{3,6}\}$, and $\{E_{3,3}, E_{3,5}, E_{3,6}\}$.

Both 2D and 3D structure creation have been studied in the literature; though some of these structures are based on grids and consist of grid-size blocks, e.g., Refs [38] and [50], others are not, e.g., Ref. [34]. Though our structures are 2D, they are special cases of grid-based 3D structures in that a 2D pattern of squares of a particular sensed type can be replaced by an equivalent set of 3D blocks of the same type positioned at the same grid-squares in the environment. Our binary conception of 2D structural damage (a 2D structural element is either present (functioning) or absent (damaged)) does not accommodate real-world structural damage, in which there are typically types and degrees of component failure; moreover, some types of real-world damage are only an issue in 3D structures, e.g., failure under loading. That being said, our basic conception of 2D grid-based damage must be handled by and, hence, is a special case of more general models of 2D grid-based damage and, as 2D grid-based structures are a special case of 3D grid-based structures by the logic above, of 3D grid-based damage as well.

2.3 Individual Robots

Our robots will be based on a finite-state architecture and, hence, will be referred to as finite-state robots (FSRs). Each robot has a compass and in a basic movement-action can either move exactly one square to the north, south, east or west of its current position or elect to stay at its current position, i.e., the set of basic movement-actions is $\{goNorth, goSouth, goWest, goEast, stay\}$. Each robot can sense for some fixed integer $r \geq 0$ the type of the square at any position within Manhattan distance r of the robot's current position (with $r = 0$ corresponding to the square on which the robot is standing); these square-types are accessible via predicates of the form $enval(e, pos)$, which returns *True* if the square at position *pos* has type $e \in E_T \cup \{e_{robot}\}$ (with the sensor returning e_{robot} if a robot is on square *pos*) and *False*, otherwise, where a position *pos* is specified in terms of a pair (x, y) specifying an environment-square $E_{i+x,j+y}$ if the robot is currently occupying $E_{i,j}$. Each robot can change the type of the square at any position within Manhattan distance $r \leq 1$ of the robot's current position to type e via predicates of the form $enmod(e, pos)$ where *pos* is specified as for $enval()$.

A robot's control-architecture is a set Q of states in which there is a special initial state q_0 and pairs of states may be linked by one or more transitions. Each such transition from a state q to a state q' has an associated activation formula f, a square-change c, and a movement-action a. Each f is either $*$ or a Boolean formula over the available sensory-predicates relative to E_T and r, and each c is either $*$ or a change-predicate. A transition is *enabled* if the robot's current state is q and f evaluates to *True*. As multiple transitions could conceivably be enabled at the same time, there are many possible ways in which FSR can operate depending on which enabled transition is chosen to execute, e.g., probabilistic, nondeterministic. For simplicity, we will restrict ourselves in this article to robots such that, at most, one transition is enabled at any time in that robot relative to that robot's current state, i.e., robots whose operation is *deterministic*. Such a robot operates as follows: Starting out from state q_0, whenever a robot is allowed to act (at arbitrary times if team operation is asynchronous and at the common clock ticks if team operation is synchronous), three rules describe what happens relative to the robot's current state q:

(1) If a single transition (q, f, c, a, q') is enabled, that transition is executed—that is, square-change c is performed if $c \neq *$, movement action a is performed, and the robot's state changes to q'.

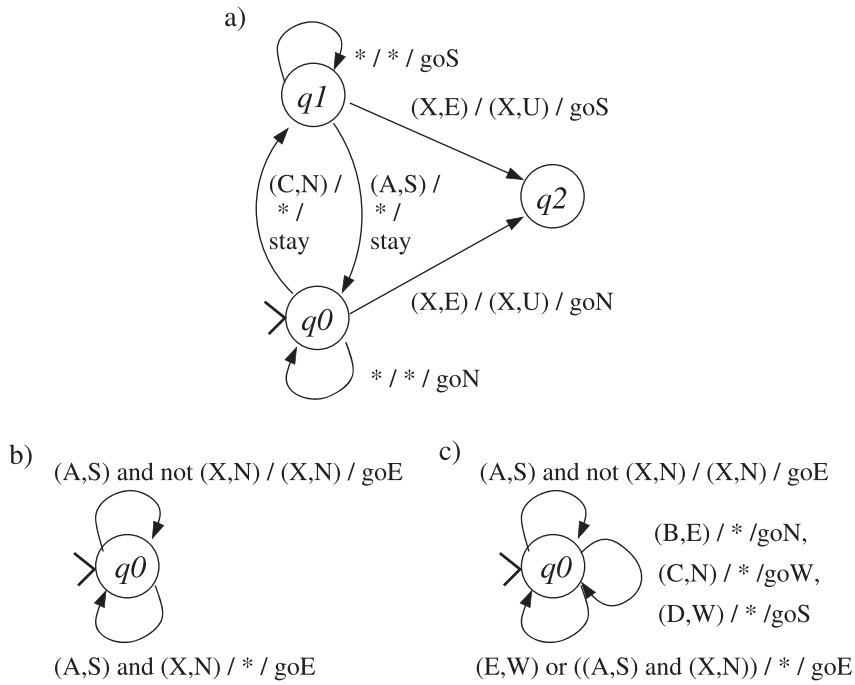

Fig. 3. Three finite-state robot controllers with $r = 1$. The initial state ($q0$) in each controller is indicated by the bold right-facing arrowhead. Each transition is labeled with a triple $x/y/z$ where x is the transition-activation formula, y is the square-change (if any), and z is the movement-action executed thereafter. As $r = 1$, there are hence at most five places at which the environment can be sensed or modified (North, South, East, West, or Underneath). Hence, *enval*(*squareType*, *pos*) and *enmod*(*squareType*, *pos*) predicates are abbreviated as (*squareType*, *direction*).

(2) If no transition is enabled, the default $*$-transition (if one has been specified relative to q) is executed.
(3) If more than one transition is enabled, the execution of the task being performed by that robot and its team is terminated.

Such robots encode a limited type of memory in their states, each of which effectively encodes a separate reactive regime of operation. In this article, we assume that sensors always perceive correctly and that movement-actions are always executed correctly.

Three finite-state robots with $r = 1$ are given in Figure 3. For example, the robot in part (a) is designed to move north (state q_0) and south (state q_1) between two southern and northern walls of square-type e_A and e_C, respectively, until it sees a structure-square of square-type e_X to its immediate east. The robot then adds a new structure-square at its current position and steps either north or south a single square depending on its current state and remains there (state $q2$). As it is very difficult in general to interpret what a robot does outside of the context of its position in its environment relative to other members of its team or the task being performed by the team, let alone decide if that robot's operation is deterministic outside of this context, the behaviors of these robots will be explained in more detail in Section 2.5.

Unlike robots proposed in the literature or used in practice, which move stochastically in a continuous manner, our robots move deterministically in a non-continuous manner. Moreover, our robot model abstracts away from issues associated with collecting and carrying construction

materials from special depots as well as the details of placing these materials on the structure. With respect to the issue of stochastic movement, we do not deny that this is a frequently-invoked solution to issues such as deadlock resolution, which appears to work well in practice. However, it has not been shown to our knowledge how frequently stochastically-operating robot teams complete their tasks relative to all possible sequences of stochastic choices, let alone how quickly this completion occurs in the worst case. Hence, given our focus in this article on guaranteed reliable and quick robot team operation (see Section 2.5), we will analyze deterministic robots and robot teams.[1] With respect to abstracting away from collecting, carrying, and placing construction materials, we do not deny that these activities are both important and difficult in themselves; rather, we have done as we do here to focus our analyses on computational difficulties associated with the core construction, repair, and maintenance processes themselves.

2.4 Robot Teams

An FSR team T is a set of FSR. Given an environment E and an FSR team T, each freespace in E can hold at most one member of T; if at any point in the execution of a task two robots in a team attempt to occupy the same freespace or a robot attempts to occupy the same space as an obstacle, the execution terminates and is considered unsuccessful.[2] A *positioning* of T in E is an assignment of the robots in T to a subset of $|T|$ squares in E. Team members do not communicate with each other directly, though they may communicate indirectly through changes they make to the environment, i.e., via stigmergy [7, 39] (see the sketch of the proof of Result B given in Section 3 for an example of such indirect communication). Team members can move either synchronously or asynchronously as specified; however, in both cases, once movement is triggered, it is instantaneous and atomic in the sense that the specified movement is completed. Given this, an asynchronous execution of T can be viewed as a sequence of executions of enabled transitions of members of T. As we have restricted ourselves to robots that operate deterministically in the sense described above, we, hence, restrict ourselves to robot teams that operate deterministically in their environments.

Unlike robot teams proposed in the literature or used in practice, which move in a continuously asynchronous manner and allow robots to communicate directly with each other, our robot teams move in the discretely synchronous or asynchronous manners described above and do not allow inter-robot communication. Note also that our robots do not have special mechanisms for avoiding or resolving collisions (in the case of both synchronous and asynchronous execution) or for ensuring that asynchronous execution is fair, e.g., that all robots with enabled transitions have the opportunity to execute those transitions; rather, these mechanisms are assumed to be either explicitly stated in or emergent properties of the individual robot controllers.

2.5 Computational Problems

Given the above, we can now formalize the computational problems associated with the construction, repair, and maintenance tasks sketched in Section 1. In formalizing these tasks, we will take into account the following requirements associated with real-world construction:

(1) Structures have specified forms, locations, and orientations.
(2) Any construction-related task must always be done correctly.
(3) Any construction-related task must always be done quickly.

[1] All this being said, it is worth noting that the deterministic finite-state controllers underlying our robots are special cases of commonly-used probabilistic finite-state controllers, in that the state-transitions in our robots always have probability of execution 1.0 if enabled (which is a special case of the more general arbitrary-value static or dynamically varying state-transition execution probabilities typical in the literature [9, Section 2.1]).
[2] Note that this corresponds to the simplest possible type of collision-avoidance policy, i.e., no collision-avoidance at all.

The first two of these requirements, if strictly adhered to, rule out not only robot teams that approximately or with some (hopefully high) probability carry out their specified tasks but also stochastic robot team design methods such as evolutionary algorithms that are only guaranteed to produce accurately-performing robot teams with some (again, hopefully high) probability. This justifies both our focus on robots based on deterministic rather than stochastic behavior rules (see Section 2.3 for more details on this choice) and deterministic robot team design algorithms. The third requirement seems reasonable, especially if the task being solved is a prelude to other tasks and must be completed prior to the execution of those tasks. We will adopt the following conceptions of fast task completion defined in Ref. [47].

Definition 2.1. For a pair of positive integers c_1 and c_2, a task is (c_1, c_2)-completable relative to a synchronous robot team T and a positioning p_I in environment E if that task can be completed by T starting at p_I in E such that the number of timesteps required by T to perform the task is upper-bounded by $c_1|E|^{c_2}$.

Definition 2.2. For a pair of positive integers c_1 and c_2, a task is (c_1, c_2)-completable relative to an asynchronous robot team T and a positioning p_I in environment E if that task is completed by each valid asynchronous enabled transition execution-sequence, which has the members of T starting at p_I and is of length upper-bounded by $c_1|E|^{c_2}$.

In the interest of simplicity, we add four more requirements of our own:

(4) All aspects of the environment E are observable and known when robot teams are designed.
(5) Robot teams are designed by selecting robots from a given library of behaviorally-distinct robots such that there may be more than one copy of a particular robot in the team.
(6) Robots start each task at a positioning in a specified region E_I of the environment that is different from that of the requested structure.
(7) Requirements 1–3 can be met relative to *any* initial positioning of the robots in the team in the initial region specified in Requirement 6.

The fourth requirement is a special case of environments that are only partially known or totally unknown at design time, and characterizes the real-world situation in which the construction site has been fully inspected to make sure both that site and available materials are appropriate for construction. The fifth requirement states that we consider here what is arguably the simplest possible type of team design. As part of this type of design, we allow the number of copies of a robot from the given library in a team to vary. This lets us generate both heterogeneous teams (in which members can have different controller-architectures and, hence, different behaviors [21, 37]) and homogeneous teams (in which all members have the the same controller-architecture and, hence, the same behavior [9, 50]).[3] The sixth requirement encodes the assumption that robots start each task at a depot distinct from the actual construction site. Finally, the seventh requirement encodes a reasonable notion of robust task execution that, relative to Requirement 6, allows robots to start in random positions in E_I (which is partway toward the typical assumption that robots start at random positions in E).

[3]Note that our simple definition of team heterogeneity will only allow different behaviors of the members of a team if the behaviors of different robots selected from the given library are different. Given that testing for behavioral equivalence is known to be intractable for many types of finite-state mechanisms [18, Section A10], for simplicity, we will assume that all pairs of robots in the given library are pairwise behaviorally distinct.

The above yields the following computational problems.

Team Design for Fast Construction (DesCon)
Input: An environment E based on square-type set E_T, a requested team-size $|T|$, a robot library L, a subset E_I of size $|T|$ of the squares in E, a structure X, a position p_X of X in E, and positive integers c_1 and c_2.
Output: A robot team T selected from L such that the task of constructing X at p_X when T starts at any positioning of the members of T in E_I is (c_1, c_2)-completable, if such a T exists, and special symbol $\perp$, otherwise.

Team Design for Fast Repair (DesRep)
Input: An environment E based on square-type set E_T, a requested team-size $|T|$, a robot library L, a subset E_I of size $|T|$ of the squares in E, a structure X positioned at p_X in E, a damage d_X to X, and positive integers c_1 and c_2.
Output: A robot team T selected from L such that the task of repairing damage d_X to X when T starts at any positioning of the members of T in E_I is (c_1, c_2)-completable, if such a T exists, and special symbol $\perp$, otherwise.

Team Design for Fast Maintenance (DesMnt)
Input: An environment E based on square-type set E_T, a requested team-size $|T|$, a robot library L, a subset E_I of size $|T|$ of the squares in E, a structure X positioned at p_X in E, and positive integers c_1 and c_2.
Output: A robot team T selected from L such that, once T starts at any positioning of the members of T in E_I, the task of repairing any damage in D_X to X on an ongoing basis is (c_1, c_2)-completable,[4] if such a T exists, and special symbol $\perp$, otherwise.

Where necessary, we add the subscripts *syn* and *asy* to the problem names to denote instances of these problems relative to synchronous and asynchronous team operation, respectively. Most of the input parameters in these problems are part of the abstract versions of these problems stated in Section 1 or are justified by the seven requirements associated with real-world construction listed earlier in this section. However, the need for others is less obvious. For example, given that one is, in general, interested in designing a team of robots that performs a specified task quickly and may not care how big the team is or precisely how long it takes to complete the task, the inclusion of parameters $|T|$, c_1, and c_2 in each of our problems may seem superfluous. However, the inclusion of these parameters allows control over and, hence, investigation of the computational effects of restricting team size and task completion speed, which is in line with our goal stated in Section 1 of fully characterizing those situations under which robot team design is and is not tractable.

Example robot-team solutions for our three problems above created relative to the environment E and structure X shown in parts (a) and (b) of Figure 2 and the library $L = \{R_a, R_b, R_c\}$ consisting of the robots shown in parts (a), (b), and (c) of Figure 3 are as follows:

—DesCon$_{syn}$: When $|T| = 4$, $E_I = \{E_{3,2}, E_{4,2}, E_{5,2}, E_{6,2}\}$, $c_1 = 3$, and $c_2 = 1$, X can be constructed by setting T to four copies of R_a.
—DesRep$_{syn}$ and DesRep$_{asy}$: When $|T| = 1$, $E_I = \{E_{3,2}\}$, d_X is any subset of $\{E_{3,2}, E_{4,2}, E_{5,2}, E_{6,2}\}$, and $c_1 = c_2 = 1$, X can be repaired by setting T to either R_b or R_c.

[4]Here, we mean that any time damage occurs on an ongoing basis, the task of repairing that damage is (c_1, c_2)-completable. We will assume that once damage occurs, repair starts immediately and no further damage occurs during the at most $c_1|E|^{c_2}$ time required to complete that repair.

—DesMnt$_{syn}$ and DesMnt$_{asy}$: When $|T| = 1$, $E_I = \{E_{3,2}\}$, and $c_1 = c_2 = 1$, X can be maintained by setting T to R_c.

In the first case, by positioning the four copies of R_a as specified, these robots will, over time, create a new wall by extending a wall westward from the wall "seed" in the original environment. In the second case, either R_b or R_c can use the wall-squares to the immediate south of the robot as a template guiding where to put wall-squares in the structure. Finally, in the third case, the extra transitions added to R_b to create R_c allow R_c to use different outer-wall square-types to circle continuously around the structure and, hence, repair any damage on an ongoing basis. Note that in all of these cases, the operation of all robots is deterministic.

Given the basic nature of our models of environments, structures, and structural damage noted previously in Sections 2.1 and 2.2, problems DesCon and DesRep abstract away from a number of issues associated with real-world construction and repair. As DesMnt extends DesRep, there are similar problems with our conception of maintenance. Moreover, DesMnt does not take into account a number of activities that are considered part of real-world structural maintenance, such as the replacement of structural elements that are faulty, the reinforcement or replacement of elements that will soon fail, or cleaning of structural elements. That being said, it was noted in Section 2.2 that our 2D grid-based models of structures and structural damage are special cases of more general 2D and 3D grid-based models of structures and structural damage; hence, our problems DesCon, DesRep, and DesMnt are special cases of more realistic problems associated with real-world construction, repair, and maintenance in the sense described in Section 1. As will be seen in Section 5, it is this relationship that will enable all intractability (and some tractability) results derived for DesCon, DesRep, and DesMnt to have a broader applicability.

3 TEAM-BASED STRUCTURE MANIPULATION IS (MOSTLY) INTRACTABLE

Let us revisit the first of the questions raised in the Introduction—namely, are there efficient design algorithms whose produced robot teams always perform construction, repair, and maintenance tasks in given environments both correctly and quickly? Following common practice in Computer Science [18], we will say that an algorithm is efficient if that algorithm solves its associated problem in polynomial time, i.e., in time upper-bounded by $c_1|x|^{c_2}$ where $|x|$ is the input size and c_1 and c_2 are constants. A problem that has a polynomial-time algorithm is *(polynomial-time) tractable*. Such algorithms are preferable to those whose runtimes are bounded by superpolynomial functions, e.g., $|x|^{\log_2 |x|}$, $2^{|x|}$, $2^{2^{|x|}}$. This is so because polynomial functions increase in value much slower than non-polynomial functions as $|x|$ gets large, which allows polynomial-time algorithms to solve much larger inputs in practical amounts of time than superpolynomial-time algorithms.

We consider first the case of homogeneous teams whose members are all the same type of robot.

Result A. DesCon$_{syn}$ and DesRep$_{syn}$ are polynomial-time tractable when teams are homogeneous.

Proof. Observe that for a robot library L, there are $|L|$ homogeneous teams that can be created from L (one for each robot in L); moreover, there is exactly one way to place each such team on the squares of E_I. Consider the following algorithm for DesCon$_{syn}$: For each robot r in L, place $|T|$ copies of r on the squares of E_I, and execute this team until either (1) X is created at p_X or (2) the execution reaches $c_1|E|^{c_2} + 1$ timesteps. If (1) occurs relative to any r in L, return the team consisting of $|T|$ copies of r; otherwise, return $\perp$. An analogous algorithm can be specified for DesRep$_{syn}$. To complete the proof, observe that both of these algorithms run in times that are upper-bounded in the worst case by functions that are polynomials of sizes of the given inputs to DesCon$_{syn}$ and DesCon$_{asy}$, respectively. □

		W	W	W	W	W	W	W	W	W	
S		***R***	***R***	***R***							
S		M1					M2	T	M3		N
E											N
			V1	V2	V3	V4					

Fig. 4. An example environment and initial placement region E_I of a FSR team T constructed in the proof of Result B when in the given instance of Dominating set, $|V| = 4$ and $k = 2$. Robot movement (N, S, E, W) and activity (V1, V2, V3, V4, M1, M2, M3, T) square-types are shown in Roman font and the initial placement of the FSRs in T is shown by bold-italic symbols R. All other squares (including those under the initial placement of the robots) have a special blank square-type.

Though we do not have similar results at this time for DesCon_{asy}, DesRep_{asy}, DesMnt_{syn}, or DesMnt_{asy} relative to homogeneous teams, the situation is much less ambiguous when teams are heterogeneous.[5]

Result B. DesCon_{syn} and DesCon_{asy} are not polynomial-time tractable when teams are heterogeneous.

Proof (Sketch). We show that DesCon_{syn} and DesCon_{asy} can be used to solve the following polynomial-time intractable problem:

Dominating set
Input: An undirected graph $G = (V, E)$ and a positive integer k.
Question: Does G contain a dominating set of size k, *i.e.*, is there a subset $V' \subseteq V$, $|V'| = k$, such that for all $v \in V$, either $v \in V'$ or there is at least one $v' \in V'$ such that $(v, v') \in E$?

Hence, there can be no polynomial-time algorithms for DesCon_{syn} or DesCon_{asy}, as such algorithms could be used to solve Dominating set in polynomial time, which would be a contradiction.

For a given instance of Dominating set, we construct an instance of DesCon_{syn} or DesCon_{asy} with an environment very similar to that in part (a) of Figure 2 such that a team of $k + 1$ robots will be able to cooperatively construct a single-square structure at the square with type T in Figure 4 if and only if there is a dominating set of size at most k in the graph in the given instance of Dominating set An example of such an environment is given in Figure 4. The robots are initially positioned in the top-left corner of the movement-track and move in a counter-clockwise fashion around this track. Robot movement and activity is dictated by squares of particular types. A functional robot team consists of at most k "vertex neighborhood" robots, which attempt to fill in a scaffolding of squares in the center of the central line of squares and at least one "checker" robot, which ensures that the squares in this scaffolding correspond to a dominating set and, if so, places the single square corresponding to the requested structure at the square with type **T** in Figure 4. Regardless of how these robots on this team are initially positioned, the vertex neighborhood robots will, if they can, have filled in the central scaffolding after the entire team has been

[5]Each polynomial-time intractability result in this section holds relative to the $P \neq NP$ conjecture, which is widely believed to be true [16, 18].

Table 1. Parameters for Team-based Structure Manipulation Problems

Parameter	Description	Value
$\|L\|$	# robots in library	3
$\|T\|$	# robots in team	4
h	# robot-types in team	1
$\|Q\|$	Max # states per robot	3
r	Max robot perceptual radius	1
$\|E\|$	# squares in environment	45
$\|E_T\|$	# distinct environment-square types	7
$\|X\|$	# squares in structure	4
$\|d_X\|$	max # squares in damage	N/A
c_1	Multiplier in task performance time	3
c_2	Exponent in task performance time	1

All values in the third column of the table are given for the first solution to the robot team design problem given at the end of Section 2.5 for the creation of the structure in part (b) of Figure 2.

around the track at most two times, leaving the checker robot to verify their work on the third time around.

Note that as the structure is a single grid square, this construction proves the polynomial-time intractability of DesCon_{syn} and DesCon_{asy} when X is the simplest possible 1D structure embedded in a 2D environment. Moreover, as the two types of robots in this construction only communicate indirectly via additions to and the sensing of the central scaffolding, this is a good example of communication by stigmergy [7, 39]. □

Given that damage in the construction above is restricted to a single square and the counter-clockwise motion of the robots around the movement-track is potentially eternal, the above also gives us the following.

Result C. DesRep_{syn} and DesRep_{asy} are not polynomial-time tractable when teams are heterogeneous.

Result D. DesMnt_{syn} and DesMnt_{asy} are not polynomial-time tractable when teams are heterogeneous.

Results B, C, and D show that neither synchronous or asynchronous heterogeneous team design for construction, repair, or maintenance in given environments can be done efficiently for all inputs. This motivates the second of the questions raised in the Introduction—namely, under what restrictions might efficient design for these tasks be possible? This will be addressed in the next section.

4 WHAT MAKES TEAM-BASED STRUCTURE MANIPULATION TRACTABLE?

To answer the question of what restrictions make team-based structure manipulation tractable, we first need to define what it means to solve a problem efficiently under restrictions. Let such restrictions be phrased in terms of aspects of our problem input or output; call each such aspect a *parameter*. Some example parameters for our problems are $|T|$ (the number of robots on a team) and r (the robot perceptual radius) (see also Table 1). A problem Π is *fixed-parameter (fp-)tractable relative to a set of parameters* $K = \{k_1, k_2, \ldots, k_m\}$ [13], i.e., $\langle K \rangle$-Π is fp-tractable, if there is an algorithm for Π whose running time is upper-bounded by $f(K)|x|^c$ for some function $f()$, where $|x|$ is the

problem input size and c is a constant. Fixed-parameter tractability generalizes polynomial-time solvability by allowing problems to be effectively solvable in polynomial time when the values of the parameters in K are small, e.g., $k_1, k_2 \leq 4$, and $f()$ is well-behaved, e.g., $1.2^{k_1+k_2}$, such that the value of $f(K)$ is a small constant. Hence, if a polynomial-time intractable problem Π is fp-tractable relative to a well-behaved $f()$ for a parameter-set K, then Π can be efficiently solved even for large inputs in which the values of the parameters in K are small.

There are many techniques for designing fp-tractable algorithms [11, 30] that have been successfully applied to a wide variety of polynomial-time intractable problems [13, 35]. Our question about efficient solvability under restrictions may now be rephrased as asking relative to which sets of parameters our problems are and are not fp-tractable. The parameters of interest in this article are shown in Table 1 and can be broken into four groups:

(1) Restrictions on robot libraries and teams ($|L|, |T|, h$);
(2) Restrictions on individual robots ($|Q|, r$);
(3) Restrictions on environments, target structures, and structure damage ($|E_T|, |X|, |d_X|$); and
(4) Restrictions on the performance time of the structure-manipulation task (c_1, c_2).

We consider first what parameters do not yield fp-tractability.[6]

Result E. $\langle |T|, h, |Q|, r, |X|, |d_X| \rangle$-DesCon$_{syn}$, -DesCon$_{asy}$, -DesRep$_{syn}$, -DesRep$_{asy}$, -DesMnt$_{syn}$, and -DesMnt$_{asy}$ are not fp-tractable.

Result F. $\langle |T|, h, |Q|, |E_T|, |X|, |d_X| \rangle$-DesCon$_{syn}$, -DesCon$_{asy}$, -DesRep$_{syn}$, -DesRep$_{asy}$, -DesMnt$_{syn}$, and -DesMnt$_{asy}$ are not fp-tractable.

These results essentially follow from the fp-intractability of $\langle k \rangle$-Dominating set and the constructions in the proofs of Results B–D sketched in Section 3. Results E and F show that neither synchronous nor asynchronous heterogeneous team design for construction, repair, or maintenance in given environments can be done efficiently under a number of restrictions. These results are more powerful than they first appear, as it is known that a problem that is fp-intractable relative to a particular parameter-set K is also fp-intractable relative to any subset of K [44, Lemma 2.1.31]. Hence, none of the parameters considered here except $|L|$ can be either individually or in many combinations be restricted to yield tractability.

Despite this, there are restrictions that do make some of our problems tractable.

Result G. $\langle |T|, |L| \rangle$-DesCon$_{syn}$ and -DesRep$_{syn}$ are fp-tractable.

Proof. Consider the following algorithm for DesCon$_{syn}$: For each possible team of $|T|$ robots that can be selected from L and each possible positioning of these robots in the squares of E_I, execute this team at that positioning until either (1) X is created at p_X or (2) the execution reaches $c_1|E|^{c_2} + 1$ timesteps. If (1) occurs relative to any examined team for all positionings of that team in E_I, return that team; otherwise, return $\perp$. An analogous algorithm can be specified for DesRep$_{syn}$. To complete the proof, observe that as the number of possible teams and positionings of those teams are upper-bounded by $|L|^{|T|}$ and $|T|! \leq |T|^{|T|}$, respectively, both of these algorithms run in times that are upper-bounded in the worst case by functions of $|T|$ and $|L|$ multiplied by functions that are polynomials of the sizes of the given inputs to DesCon$_{syn}$ and DesRep$_{syn}$, respectively. □

Result H. DesCon$_{syn}$ and DesRep$_{syn}$ are polynomial-time tractable when $|T|$ is a constant with value ≥ 1.

[6]Each fp-intractability result in this section holds relative to conjecture $FPT \neq W[1]$, which is widely believed to be true [13].

Proof. This follows from the algorithms in the proof of Result G and the observation that the $|L|^{|T|}$ and $|T|!$ terms in the runtimes of these algorithms collapse to a polynomial function of $|L|$ and a constant when $|T|$ is a constant of value ≥ 1. □

Result I. $\langle |T|, |Q|, r, |E_T| \rangle$-DesCon$_{syn}$ and -DesRep$_{syn}$ are fp-tractable.

Proof. As each robot in L is behaviorally distinct (see Footnote 3), let us upper-bound the maximum number of behaviorally-distinct robots and, hence, $|L|$ by a function of $|Q|$, r, and $|E_T|$. This will be done by computing upper bounds on the following four quantities:

(1) *The number of squares that a robot with sensory radius r can sense*: This is less than $(2r + 1)^2$; call this quantity q_1 and let S_1 be the set of these squares.
(2) *The number of world-configurations that can be sensed by a robot with sensory radius r*: As each sensed square either has a square-type from E_T or a robot, this is less than $(|E_T| + 1)^{q_1}$; call this quantity q_2 and let S_2 be the set of these world-configurations.
(3) *The number of distinct transition-configurations of a state q in a $|Q|$-state FSR*: Each class in an ordered partition of S_2 (or rather, the conjunction of the world-configurations in that class) specifies the activation-formula of a transition outward from q to another state in the FSR; the number of such partitions is equal to the q_2st ordered Bell number, which can be approximated as $\frac{q_2!}{2(\log 2)^{q_2+1}}$. Observe that the number of classes in each such partition is less than or equal to q_2. Each transition has an associated movement-direction (selected from the set $\{goNorth, goSouth, goEast, goWest, stay\}$), square-type modification request (consisting of either (1) one of the nine squares within radius 1 of the robot's current position and a square-type from E_T or (2) $*$), and next-state destination (selected from the set $\{1, 2, \ldots, |Q|\}$). Hence, the number of distinct transition-configurations of a state q in a $|Q|$-state FSR is thus less than $\frac{q_2!}{2('log2)^{q_2+1}} 5^{q_2} (9 \times (|E_T| + 1))^{q_2} |Q|^{q_2}$; call this quantity q^3 and let S_3 be the set of these transition-configurations.
(4) *The number of behaviorally-distinct robots with $|Q|$ states and sensory radius r*: Each robot must choose a transition-configuration for each of its $|Q|$ states from S_3. Hence, the number of behaviorally-distinct robots with $|Q|$ states and sensory radius r is less than $q_3^{|Q|}$; call this quantity q_4.

As $|L| < q_4$, to complete the proof, observe that both of the algorithms for DesCon$_{syn}$ and DesRep$_{syn}$ in the proof of Result G run in times that are upper-bounded in the worst case by functions of $|T|$, $|Q|$, r, and $|E_T|$ multiplied by functions that are polynomials of the sizes of the given inputs to DesCon$_{syn}$ and DesRep$_{syn}$, respectively. □

Again, Results G and I are more powerful than they first appear, as it is known that a problem that is fp-tractable relative to a particular parameter-set K is also fp-tractable relative to any superset of K [44, Lemma 2.1.30]. Hence, any set of parameters including both $|T|$ and $|L|$ or all of $|T|$, $|Q|$, r, and $|E_T|$ can be restricted to yield tractability for synchronous team design for construction and repair.

5 GENERALITY OF RESULTS

A valid objection to our results above is that, as they were derived relative to simplified models of 2D structure construction, repair, and maintenance, they are of very limited use. However, it turns out that these results have a broad applicability because our models are special cases of a number of more realistic models,

- Our 2D grid-based environments are special cases of 3D grid-based environments (see Section 2.1 for details).
- Our fully-known 2D grid-based environments are special cases of partially known and totally unknown 2D and 3D grid-based environments (as the class of all such environments include fully known environments as a special case).
- Our model of 2D grid-based structures is a special case of more realistic models of 2D and 3D grid-based structures (see Section 2.2 for details).
- Our model of 2D grid-based structural damage is a special case of more realistic models of 2D and 3D grid-based structural damage (see Section 2.2 for details).
- Our deterministic FSR model is a special case of probabilistic FSR models (see Footnote 1 for details).
- Our exact model of FSR motion and sensing is a special case of probabilistic models allowing imprecise FSR motion and sensing (as the class of all such models includes the model with exact motion and sensing as a special case).
- Our team operation model, which does not allow direct communication between robots, is a special case of team operation models that do allow some form or degree of direct communication between robots (as the class of all such models includes the model with no direct communication as a special case).

More realistic versions of DesCon, DesRep, and DesMnt can be created by replacing any combination of simple special-case models with the more general models above. For example, one could define a version of DesRep that repairs more realistic types of damage to 3D grid-based structures using teams of probabilistic FSRs that can communicate with each other. Courtesy of the special-case relationship described in Section 1, any automated system that solves such a more realistic problem Π' can also solve the original problem Π defined relative to the simple special-case models analyzed here. Depending on the nature of the special-case relationship, this will sometimes involve recoding problem entities, e.g., translating 2D grid-based environments and structures into equivalent 3D grid-based environments and structures by replacing squares of particular types with blocks of those types along the lines sketched in Sections 2.1 and 2.2. Intractability results for Π must then also apply to Π' as well as the operation of any automated system solving Π'. To see this, suppose Π is intractable; if Π' is tractable by algorithm A, then A can be used to solve Π efficiently, which contradicts the intractability of Π. Hence, Π' must also be intractable. This is stated more formally in the following observations from Ref. [47]:

Observation 1. *Any polynomial-time intractability result for a problem* Π *also applies to any problem* Π' *that has* Π *as a special case.*

Observation 2. *Any fp-intractability result for a problem* $\langle K \rangle$-Π *also applies to* $\langle K \rangle$-Π' *for any problem* Π' *that has* Π *as a special case.*

Observation 3. *If a problem* Π *is polynomial-intractable when a parameter* p *of* Π *is of constant value* c*, then* Π *cannot have an algorithm that, for some specified constant* $c'' \geq c$*, runs correctly in* $f(p)|x|^{c'}$ *time for a constant* c' *and any input* x *in which* $p \leq c''$*.*

By exploiting the details of our intractability proofs, these observations can yield additional, more specific results, e.g.,

- As all of our intractability results hold for the simplest possible target structures, i.e., a single 2D square, and 2D grid-based structures are special cases of grid-based 3D structures, by Observations 1 and 2, these results apply to structure-manipulation team design relative to *all* possible 2D *and* grid-based 3D target structures.

Table 2. A Detailed Summary of Our Parameterized Complexity Results

	DesCon				DesRep				DesMnt	
	E	F	**G***	**I***	E	F	**G***	**I***	E	F
$\|L\|$	–	–	**@**	**–**	–	–	**@**	**–**	–	–
$\|T\|$	@	@	**@**	**@**	@	@	**@**	**@**	@	@
h	@	@	**–**	**–**	@	@	**–**	**–**	@	@
$\|Q\|$	3	3	**–**	**@**	3	3	**–**	**@**	3	3
r	1	–	**–**	**@**	1	–	**–**	**@**	1	–
$\|E_T\|$	–	12	**–**	**@**	–	12	**–**	**@**	–	12
$\|X\|$	1	1	**–**	**–**	1	1	**–**	**–**	1	1
$\|d_X\|$	N/A	N/A	N/A	N/A	1	1	**–**	**–**	1	1
c_1	1	1	**–**	**–**	1	1	**–**	**–**	1	1
c_2	1,2	1,2	**–**	**–**	1,2	1,2	**–**	**–**	1,2	1,2

Each column in this table is a result holds relative to the parameter-set consisting of all parameters with a @-symbol in that column. If the result holds when a particular parameter has a constant value c, that is indicated by c replacing @ for which parameter in that result's column. The two constants listed for parameter c_2 denote values of c_2 which hold for results relative to synchronously and asynchronously operating robot teams, respectively. Results are grouped by problem, with fp-intractability results first and fp-tractability results (shown in bold) last. Results only hold relative to synchronous robot team operation are denoted by star (*) superscripts.

—As DesCon_{syn}, DesCon_{asy}, DesRep_{syn}, DesRep_{asy}, DesMnt_{syn}, and DesMnt_{asy} are polynomial-time intractable when each of the parameters $|Q|$, r, $|E_T|$, $|X|$, and $|d_X|$ have small (and in the case of $|X|$ and $|d_X|$, the smallest possible) constant values (see Table 2 and the detailed statements of these results in the Appendix), by Observation 3, these problems cannot have fixed-parameter-like algorithms of the form described in Observation 3 that effectively operate in polynomial time relative to larger values of these parameters.[7]

Readers interested in other such specific results, possibly relative to other parameters of interest, are encouraged to try deriving them relative to our (or their own) intractability proofs.

Algorithms often exploit particular details of the inputs and outputs in their associated problems to attain efficiency or even work at all, so tractability results typically do not propagate from special cases to more general problems. That being said, our tractability results also have a surprisingly broad applicability relative to points 4–6 above, as long as details relating to robot controller model and initial team positionings in the environment remain as defined in Sections 2.3 and 2.4. This is because the algorithms described in the proofs of Results G–I depend only on the combinatorics limiting the number of possible robot teams and initial team positionings.

The above does, as promised at the beginning of this section, establish that both our intractability and tractability results have a broad applicability. However, this applicability is still limited to systems in which robot teams move in a non-continuous deterministic manner in grid-based

[7]A natural question at this point is whether there are specific constant values of these parameters under which efficient verification and design is possible, e.g., $|Q| = 5$ or $r = 3$. This is not ruled out by Observation 3, which only considers algorithms that work for all parameter-values less than or equal to (rather than strictly equal to) a specified value. However, in certain cases, our intractability results can still hold by modifying our proof constructions to create "padded" inputs in which the values of the parameters are artificially raised to higher constant values. For example, one can have intractability when $|Q| = c$ for any constant $c > 3$ using teams in which all robots have extra states and associated transitions that are never executed.

environments. In the next section, we will discuss how we can extend this applicability to real-world robotic systems.

6 DISCUSSION

What does all the above ultimately mean to practitioners of distributed robotics? Given that the type of team design investigated in this article (namely, selection of team members from a provided library) has not been studied to date in the literature and, perhaps, more importantly, all our results are derived relative to a model in which simple robots move in a non-continuous deterministic manner in grid-based environments, it may initially seem like not much at all. However, if one sees this article as the start of a (hopefully, ongoing) conversation between robotics and computational complexity theory about possible (and not just bio-inspired) robot team design algorithms, there is potentially much of value. Accordingly, in this section, we will first discuss the implications of our results for the simplified systems that we analyzed (Section 6.1) and then discuss how such results and others like them could be used in real-world robotics (Section 6.2).

6.1 Implications for Simplified Robot Team Design

Our general intractability results (Results B–D) show that our construction, repair, and maintenance problems cannot be solved efficiently in general. To the extent that our problems defined and analyzed in this article are special cases of real-world problems (see Section 5), these results hint that difficulties encountered to date in deriving robot team design algorithms for construction that are both efficient and reliable are not only to be expected but may also be unavoidable. Our parameterized intractability results (Results E and F), in turn, show that surprisingly few restrictions on robot controllers, environments, and target structures yield tractability, even if synchronous and asynchronous robot teams are required to complete their tasks in linear and quadratic time, respectively (see results relative to c_1 and c_2 in Table 2). It is unfortunate that our parameterized tractability results (Results G–I), being tied to details of our design model, are not similarly enlightening. Nonetheless, they do suggest types of restriction-combinations that may yield tractability relative to more realistic team design problems and, hence, may serve for now as promissory notes on practical algorithms that will be developed in the future (see Section 6.2).

Intriguing as our current results are, they are incomplete—that is, though these results characterize the parameterized complexity of our problems relative to many combinations of the parameters in Table 1, they do not do so for all and, hence, only yield a partial intractability map (see Section 1). Viable robot team design algorithms may thus still be lurking relative to these uncharacterized combinations and need to be either confirmed or discounted. A first step in this would be to see if fp-tractability holds relative to subsets of the parameter-sets $\{|T|, |L|\}$ and $\{|T|, |Q|, r, |E_T|\}$ in Results G and I; Result H already suggests that the former may be possible. It would also be of great interest to know if parameterized intractability holds for constant values of $|L|$ and h, the only two of our parameters for which this is not currently so. The case of h is particularly interesting, as it is already known courtesy of Result A that $h = 1$ grants polynomial-time solvability in certain cases; intractability results for constant h (in the best case, for $h = 2$) would very much help in clarifying the extent to which team heterogeneity affects the computational difficulty of the design process (and, hence, may justify the common focus in the literature on homogeneous robot teams).

Last, but certainly not least, we also need to derive tractability results for team design problems for construction and repair for asynchronous teams as well as team design problems for maintenance. This has been made very difficult by our models of asynchronous operation (which seems to involve looking at all $|T|^{c_1|E|^{c_2}}$ possible enabled transition-execution sequences for a given team in the worst case) and maintenance (which seems to require reasoning about team behavior in all

possible futures after the start of operations, even in the case of synchronous teams). Whether these difficulties are resolvable relative to our current models and problems, or require modifications of these models and problems, remains to be seen.

6.2 Implications for Real-World Robot Team Design

There are essentially three difficulties with applying the results in this article to real-world distributed robotics:

(1) The algorithms underlying our fixed-parameter tractability results have impractical runtimes;
(2) Even if we can improve the runtimes of these algorithms, they (and our intractability results) are derived relative to simplified models of robots and environments; and
(3) Even if these analyses can be redone relative to more realistic models, it is possible that such models will still simplify aspects of the real world that will cause robot teams derived and optimized relative to these models to perform much worse than these models suggest they will when implemented in real-world hardware.

The last of these is the long-recognized "Reality Gap" between the performance of simulated and real-world robots [10, 24]. However, the other two are also Reality Gaps of different types, and all three need to be dealt with in at least a minimally acceptable manner.

The first gap is in artifact of the systematic parameterized complexity analysis done in this article and may be the easiest to deal with. Recall from Section 1 that the goal in doing such analyses is to provide a general overview of the options for restricting problems to obtain practical algorithms, and that the resulting overviews are useful in establishing an engineering-like framework for subsequent algorithm use and development. In this article, we have presented a partial overview for various construction-associated robot team design problems; it is the job of future research to derive the best possible algorithms relative to those sets of restrictions for which we know algorithms exist. There are a number of established techniques for deriving such algorithms [11, 30], and it has been observed multiple times within the parameterized complexity community that once fixed-parameter tractability is established, these techniques are applied by different groups of researchers in "FPT Races" to produce increasingly (and, on occasion, spectacularly) more efficient algorithms [27]. For example, once it was realized in 1992 that the polynomial-time intractable problem Vertex Cover is fixed-parameter tractable relative to the parameter k encoding the size of the wanted vertex cover, the $2^k k^{2k+2} + k|V|$ runtime of the initial fixed-parameter algorithm was bettered over the next 20 years to $1.2738^k + k|V|$. It seems reasonable to conjecture that, if there is sufficient interest, such an improvement could also happen with respect to the fixed-parameter tractability results in this article.

The second gap is an artifact of the simplifications made in computational complexity analysis, in particular, and mathematical analysis, in general, when one is trying to model phenomena involving physical reality. As noted previously in Ref. [47], such simplifications are typically made to allow analysis to be done at all, and it is common for analysis to start with simple models and then (by using the insights obtained from as well as the techniques developed to perform these analyses) progress to more realistic models. Given that intractability results for simplified models also apply to models that have such simplified models as special cases (as pointed out in Section 1 and discussed at greater length in Section 5), simplicity is a useful guideline in the initial stage of computational complexity analysis. As previously noted in Refs [47] and [48], this is not to say that extending analyses such as those given in this article to more complex models incorporating more realistic (in particular, continuous) robot dynamics and operating environments will be easy or that these analyses will also find that intractability is as widespread as we have found it to be

relative to simplified models. The computational power of real-world physical systems is the subject of vigorous ongoing debate [1] and there are many examples of problems where allowing entities to be continuous rather than discrete does (e.g., linear [25] vs. 0/1 integer [18, Problem MP1] programming) and does not (e.g., finding Steiner trees in graphs [18, Problem ND12] vs. finding Steiner trees in the 2D Euclidean plane [18, Problem ND13]) cause computational complexity to decrease. However, there is no reason to expect that such extended analyses cannot be done, and it is our hope that at least some of the proof techniques we have developed here will be of use in this endeavor.

The third gap is perhaps the most problematic for applying computational complexity analysis in a manner useful to real-world distributed robotics, much as it has been in evolutionary robotics [6, 28]. This gap may be unbridgeable if it arises from key aspects of real-world robotics that are simultaneously hard to model and analyze accurately. However, if we are lucky, all key aspects can be modeled and analyzed accurately, and the remaining gap between model and reality is small enough that it can be accommodated in some manner, e.g., the added "envelope of noise" in Jakobi's minimal simulation method [23]. The key issue here is isolating those key aspects of the model that can be modeled and analyzed accurately. It seems very likely that experiments involving real-world robots will be invaluable in doing this. For example, just as so-called robot-in-the-loop simulation optimization seems to be the best technique developed to date for dealing with the Reality Gap in evolutionary robotics [28, Section 2], a process in which rounds of model revision, complexity analysis, and design algorithm derivation alternate with rounds of implementing and testing derived algorithms in real-world robots may be the best way of arriving not only the most accurate robot model for analysis but also the most efficient and reliable robot team algorithms possible for implementation. The precise form of this process is not yet clear, i.e., whether the complexity analysis invoked is purely classical computational or parameterized complexity analysis (as is the case in the language [32] and cognition [42] complexity games for deriving the most accurate and efficient models of specified natural language and cognitive abilities, respectively). However, given its promise, developing and applying such a process is probably one of the most important directions for future research arising out of that described in this article.

7 CONCLUSIONS AND FUTURE RESEARCH

In this article, we have given the first computational and parameterized complexity analyses of several problems associated with the design of robot teams by selection from a robot library for creating, repairing, and maintaining target structures in given environments. These analyses have been done relative to a simple finite-state robot controller model defined in Ref. [47] that moves in a non-continuous deterministic manner in a grid-based environment. As such, these analyses complement and extend analyses done in Refs [41] and [47]. We have shown that all of our problems are polynomial-time intractable in general relative to both synchronous and asynchronous heterogeneous robot teams, and that they remain intractable under a number of plausible restrictions (both individually and in many combinations) on robot controllers, environments, and target structures. We have also given the first restrictions relative to which some of these problems are tractable as well as a discussion of how theoretical tractability and intractability results like those derived here can be combined with physical robot experiments to derive the best possible algorithms for real-world robot team design.

There are a number of promising directions for future research. Aside from work sketched in Section 6.1 associated with completing the analysis started in this article, this analysis should also be extended to consider new parameters. Of particular interest are parameters based on more detailed characterizations of environments and structure-manipulation tasks, such as the amounts of structure either initially in the environment or subsequently created by the robot team that are

not part of the target structure but are instrumental in manipulating that structure. These would be of great help in quantifying and investigating environmental templates and stigmergic robot interaction, respectively, as well as tradeoffs between the two [7, 39]. First steps in this have been made in Ref. [40]. It is also critical that more realistic problems be formalized and investigated. Of particular interest here is more realistic notions of robot and robot team operation (continuous asynchronous motion in a continuous 3D environment) as well as more realistic conceptions of 3D structure repair and maintenance that allow more types and degrees of damage as well as more types of repair and maintenance activities. It would also be useful to examine the complexity implications of allowing stochastic robots and robot team operation, to better assess the relative advantages of deterministic and stochastic robot teams as well as the nature of the tradeoffs (if any) between the reliability and speed of task completion under both modes of robot team operation.

The most important direction for future work, though, is the development of practical algorithms for construction-related robot team design. This will involve tackling the three types of "reality gaps" discussed in Section 6.2 between the robot team design models that are analyzed using computational complexity analysis and real-world robots and construction tasks. It is our hope that the techniques and analyses given in this article will eventually be of use in deriving the best possible practical algorithms for team-based structure creation, repair, and maintenance as well as other problems arising in distributed robotics.

APPENDIX

A PROOFS OF INTRACTABILITY RESULTS

All of our intractability results will be derived using polynomial-time and parameterized reductions. Given two problems Π and Π', a polynomial-time reduction from Π to Π' [18] is a polynomial-time algorithm for transforming instances of Π into instances of Π' such that any polynomial-time algorithm for Π' can be used in conjunction with this instance transformation algorithm to create a polynomial-time algorithm for Π. Analogously, a parameterized reduction from $\langle K \rangle$-Π to $\langle K' \rangle$-Π' [13] allows the instance transformation algorithm to run in fp-time relative to K and requires the following: (1) for each $k' \in K'$, there is a function $g_{k'}()$ such that $k' = g_{k'}(K)$ and (2) the instance transformation algorithm can be used in conjunction with any fp-algorithm for $\langle K' \rangle$-Π' to create an fp-algorithm for $\langle K \rangle$-Π. The power of such reductions comes from the following observation: given a reduction from a problem Π to a problem Π' that preserves a particular type of tractability, if Π is not tractable, then neither is Π' (otherwise, if Π' was tractable by some algorithm A, A could be combined with the instance transformation algorithm in the reduction to prove Π tractable, which is a contradiction). If necessary, the "seed" intractable problem Π used in such a proof may only be known to be intractable if a particular conjecture holds, e.g., $P \neq NP$, $FPT \neq W[1]$; in those cases, the reduction-derived intractability result for Π' holds relative to that conjecture.

All of our reductions are from the problem Dominating set given in the sketched proof of Result B in Section 3. This problem is *NP*-hard in general [18, Problem GT2] and $W[2]$-hard relative to parameter-set $\{k\}$ [13]. For each vertex v in the vertex-set V of graph G, let the complete neighborhood $N_C(v)$ of v in G be the set composed of v and the set of all vertices in G that are adjacent to v by a single edge, i.e., $v \cup \{u \mid u \in V \text{ and } (u, v) \in E\}$. We assume below for each instance of Dominating set an arbitrary ordering on the vertices of V such that $V = \{v_1, v_2, \ldots, v_{|V|}\}$.

For technical reasons, all intractability results are proved relative to decision versions of problems, i.e., problems whose solutions are either "yes" or "no". Though none of our problems defined in Section 2.5 are decision problems, each can be made into a decision problem by asking if that problem's requested output exists; let that decision version for a problem $\mathbf{X}$ be denoted by $\mathbf{X}^D$. The

following easily-proven lemma will be useful below in transferring results from decision problems to their associated non-decision problems; it follows from the observation that any algorithm for non-decision problem **X** can be used to solve $\mathbf{X}^D$.

LEMMA A.1. *If* $\mathbf{X}^D$ *is NP-hard, then* **X** *is not solvable in polynomial time unless* $P = NP$.

LEMMA A.2. DOMINATING SET *polynomial-time reduces to* $DesCon^D_{syn}$ *such that in the constructed instance of* $DesCon^D_{syn}$, $|Q| = 3$, $r = |X| = c_1 = c_2 = 1$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* DOMINATING SET.

PROOF. Given an instance $\langle G = (V, E), k\rangle$ of DOMINATING SET, construct an instance $\langle E', E_T, |T|, L, E_I, X, p_X, c_1, c_2\rangle$ of DesCon^D_{syn} as follows: Let E' be a $(|V| + 8) \times 5$ grid composed of squares of the types $E_T = \{e_N, e_S, e_E, e_W, e_{v1}, e_{v2}, \ldots e_{v|V|}, e_B, e_T, e_{M1}, e_{M2}, e_{M3}, e_X\}$ as follows:

- $E'_{1,2} = e_E$.
- $E'_{|V|+8,2} = E'_{|V|+8,3} = e_N$.
- For $3 \leq i \leq (|V| + 7)$, $E'_{i,5} = e_W$.
- $E'_{1,3} = E'_{1,4} = e_S$.
- $E'_{3,3} = e_{M1}$.
- $E'_{|V|+4,3} = e_{M2}$.
- $E'_{|V|+5,3} = e_T$.
- $E'_{|V|+6,3} = e_{M3}$.
- For $1 \leq i \leq |V|$, $E'_{i+3,1} = e_{vi}$.
- All remaining squares in E' have type e_B.

Let $|T| = k + 1$, $E_I = \{E'_{j,4} \mid 3 \leq j \leq k + 3\}$. An example environment E and initial placement region E_I of an FSR team T when $|V| = 5$ and $k = 2$ is shown in Figure 4. Robot Library $L = \{r_{v1}, r_{v2}, \ldots, r_{v|V|}, r_{chk}\}$ consists of $|V|$ vertex neighborhood robots r_{vi}, $1 \leq i \leq |V|$, and a checker robot r_{chk}, all with sensory radius $r = 1$. These robots are specified as follows:

- **Vertex neighborhood robot**: Each vertex neighborhood robot r_{vi} is based on a single state q_0 and has the following transitions:
  1. $\langle q_0, (enval(e_E, (-1, 0))$ or $enval(e_{M1}, (0, 1))$ or $enval(e_{M2}, (0, 1))$ or $enval(e_T, (0, 1))$ or $enval(e_{M3}, (0, 1))$ or $enval(e_X, (0, 1)))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_0\rangle$.
  2. $\langle q_0, enval(e_N, (1, 0)$ and not $enval(e_{robot}, (0, 1)), *, goNorth, q_0\rangle$.
  3. $\langle q_0, enval(e_W, (0, 1))$ and not $enval(e_{robot}, (-1, 0)), *, goWest, q_0\rangle$.
  4. $\langle q_0, enval(e_S, (-1, 0)$ and not $enval(e_{robot}, (0, -1)), *, goSouth, q_0\rangle$.
  5. For $v_j \in N_C(v_i)$, $\langle q_0, (enval(e_{vj}, (0, -1))$ and not $enval(e_X, (0, 1)))$ and not $enval(e_{robot}, (1, 0)), enmod(e_X, (0, 1)), goEast, q_0\rangle$.
  6. For $v_j \in N_C(v_i)$, $\langle q_0, (enval(e_{vj}, (0, -1))$ and $enval(e_X, (0, 1)))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_0\rangle$.
  7. For $v_j \notin N_C(v_i)$, $\langle q_0, enval(e_{vj}, (0, -1))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_0\rangle$.

  The transitions in (2), (3), and (4) allow r_{vi} to progress northward along around the west, westward along the north, and southward along east sides of the movement-track, respectively. All remaining transitions cover movement on the south side of the movement-track. The transitions in (5) allow r_{vi} to put a square of type e_X in the scaffolding to its immediate north and progress eastward if there is a square to its immediate south whose type corresponds to a vertex in the complete neighborhood of v_i in G, while the transitions in (6) and (7) allow eastward progression if either such an e_X is already in place or the type

of the square to the immediate south corresponds to a vertex that is not in the complete neighborhood of v_i in G. Finally, the transition in (1) allows r_{vi} to progress eastward over all remaining positions on the south side of the movement-track.

— **Checker robot**: The checker robot r_{chk} is based on three states q_0, q_1, and q_2 and has the following transitions:

(1) $\langle q_0, enval(e_E, (-1, 0))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_0\rangle$.
(2) $\langle q_0, enval(e_N, (1, 0))$ and not $enval(e_{robot}, (0, 1)), *, goNorth, q_0\rangle$.
(3) $\langle q_0, enval(e_W, (0, 1))$ and not $enval(e_{robot}, (-1, 0)), *, goWest, q_0\rangle$.
(4) $\langle q_0, enval(e_S, (-1, 0))$ and not $enval(e_{robot}, (0, -1)), *, goSouth, q_0\rangle$.
(5) $\langle q_0, enval(e_{M1}, (0, 1))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_1\rangle$.
(6) $\langle q_1, enval(e_X, (0, 1))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_1\rangle$.
(7) $\langle q_1, enval(e_{M2}, (0, 1))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_1\rangle$.
(8) $\langle q_1, enval(e_T, (0, 1))$ and not $enval(e_{robot}, (1, 0)), enmod(e_X, (0, 1)), goEast, q_1\rangle$.
(9) $\langle q_1, enval(e_{M3}, (0, 1))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_0\rangle$.
(10) $\langle q_1, enval(e_B, (0, 1))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_2\rangle$.
(11) $\langle q_2, (enval(e_X, (0, 1))$ or $enval(e_B, (0, 1))$ or $(enval(e_{M2}, (0, 1))$ or $enval(e_T, (0, 1)))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_2\rangle$.
(12) $\langle q_2, enval(e_{M3}, (0, 1))$ and not $enval(e_{robot}, (1, 0)), *, goEast, q_0\rangle$.

The transitions in (2), (3), and (4) allow r_{chk} to progress northward along the west, westward along the north, and southward along the east sides of the movement-track, respectively. All of this movement is done when r_{chk} is in state q_0. All remaining transitions cover movement on the south side of the movement-track. On entering the south side, r_{chk} enters state q_1 (transition in (5)) and remains there if the scaffolding has been completely filled in by the vertex neighborhood robots (transitions in (6–7)). This then allows r_{chk} to create X at p_X (transition in (8)) and then returns to state q_0 (transition in (9)). If at any point r_{chk} finds that part of the scaffolding has not been filled in (transition in (10)), it enters state q_2, skips over the remainder of the scaffolding (transition in (11)) and then returns to state q_0 (transition in (12)). Finally, the transition in (1) allows r_{chk} to progress eastward over all remaining positions on the south side of the movement-track.

Finally, let X be a single square of type e_X at position $E'_{|V|+5,3}$, and $c_1 = c_2 = 1$. Note that this instance of DesCon^D_{syn} can be constructed in time polynomial in the size of the given instance of Dominating set.

We now need to show the correctness of the reduction described above by proving that there is a dominating set of size at most k in G in the given instance of Dominating set if and only if there is robot team T consisting of $k + 1$ robots from L in the constructed instance of DesCon^D_{syn} such that, when started at any positioning of the team members in E_I, the task of constructing X at p_X is (c_1, c_2)-completable. The following easily-verifiable observations will be useful in this proof:

(1) Once placed on the movement-track, both of the types of robots defined above remain on that track and move in a counter-clockwise manner around that track.
(2) As at most one transition can be enabled in either of these types of robots when it is anywhere on the movement-track and in any relation to other robots on that track, the operation of T in E is deterministic.
(3) No robot can collide with another robot, as no transition can enable a robot to move in a particular direction if there is a robot already occupying the square in that direction.

(4) When operating synchronously, k timesteps after the robots in T start executing their construction task, all robots in T will be spaced exactly one square apart around the movement-track in E'.
(5) A vertex neighborhood robot r_{vi} will place a square of type e_X two squares to the north of any squares of type e_{vj} if and only if $v_j \in N_C(v_i)$.
(6) A checker robot will place a square of type e_X at $E'_{|V|+5,3}$ if and only if both (1) the robot has already encountered a square of type e_{M1} to the immediate north (so that the robot can enter state q_1) and (2) all squares in the center line from $E'_{4,3}$ to $E'_{|V|+3,3}$ are of type e_X.

Such a proof of correctness is required so that any polynomial-time algorithm for DesCon^D_{syn} can be run on the output of the instance transformation algorithm in the reduction above to yield a polynomial-time algorithm for Dominating set.

Our proof of correctness will be done in two parts. First, suppose that there is a $V' \subseteq V$, such that $V' = \{v'_1, v'_2, \ldots, v'_k\}$ is a dominating set of size k in G. Let $T = \{r_{v'1}, r_{v'2}, \ldots, r_{v'k}, r_{chk}\}$. Observe that regardless of the positions of the members of T in E_I, after at most $k + |V| + 10$ timesteps, all vertex neighborhood robots have progressed around the track to pass the square with type e_{T2}; moreover, in at most $2|V| + 16$ timesteps after that, the checking robot will have had a chance to examine all squares in the central line between the squares with types e_{M1} and e_{M2} to see if they have been set to e_X by the vertex neighborhood robots and, if so, to reach the square with the type e_T and change it to type e_X. As $k \leq |V|$, $(k + |V| + 10) + (2|V| + 16) \leq 4|V| + 26 < 5 \times (|V| + 8) = 5|V| + 40 = |E|$, the number of timesteps required for T to construct X at p_X is $\leq c_1|E|^{c_2} = |E|$, which means that this construction task is (c_1, c_2)-completable w.r.t. T when $c_1 = c_2 = 1$.

Conversely, suppose that there is robot team T consisting of $k + 1$ robots from L such that, when started at any positioning of the team members in E_I, the task of constructing X at p_X is (c_1, c_2)-completable. In order to construct X at p_X, at least one instance of robot r_{chk} must be in T (as this is the only kind of robot in L that can create squares of type e_X). Such an r_{chk} can only create a square of type e_X if that robot is in state q_1 and this can only happen if all squares between the squares with types e_{M1} and e_{M2} in the central line have type e_X. However, as checker robots cannot place squares of type e_X in this portion of the central line, the remaining vertex neighborhood robots in T must have been able to place all those e_X-squares. This would then mean that the vertices in G corresponding to those vertex neighborhood robots form a dominating set of size at most k in G (which can be padded with arbitrary additional vertices if necessary to create a dominating set of size k in G).

To complete the proof of this result, note that in the constructed instance of DesCon^D_{syn}, $|Q| = 3$, $r = |X| = c_1 = c_2 = 1$, and $|T| = h = k + 1$. □

Lemma A.3. Dominating set *polynomial-time reduces to* DesCon^D_{syn} *such that in the constructed instance of* DesCon^D_{syn}, $|Q| = 3$, $|E_T| = 12$, $|X| = c_1 = c_2 = 1$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Observe that each square-type e_{vi} in the reduction given in the proof of Lemma A.2 above can be replaced by a sequence of length $\lceil \log_2 |V| \rceil$ composed of squares of type e_0 and e_1 corresponding to a binary encoding of the integer i. If these sequence-encodings (oriented as north-south columns) replace the e_{vi}-type squares in E', we now have an environment E' based on a $(|V| + 8) \times (\lceil \log_2 |V| \rceil + 4)$ grid; we must also modify the vertex neighborhood robot to have sensory radius $r = \lceil \log_2 |V| \rceil$ and modify all transition-activation formulas in these robots to recognize the appropriate column-encodings of the e_{vi}-squares. Given this, the remainder of the reduction including the proof of correctness works as in the proof of Lemma A.2. To complete the

proof, note that in the constructed instance of DesCon^D_{syn}, $|Q| = 3$, $|E_T| = 11$, $|X| = c_1 = c_2 = 1$, and $|T| = h = k + 1$. □

Lemma A.4. Dominating set *polynomial-time reduces to* DesCon^D_{asy} *such that in the constructed instance of* DesCon^D_{asy}, $|Q| = 3$, $r = |X| = c_1 = 1$, $c_2 = 2$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Consider the reduction in Lemma A.2. If the robots in this reduction operate asynchronously, it will take in the worst-case $2k(2|V| + 12)$ enabled transition-executions for a particular robot in T to complete a circuit of the movement track back to its original position. This is so because in the worst case, the robot at the front of the team will execute transitions required to move to the back of the team before another robot moves; as this back-position is k squares short of where that lead robot was originally, it must wait for all other robots in the team to move before it can go around again to complete the circuit to its original position. Observe that the checker robot can only evaluate if the vertex neighborhood robots have completed the central line of e_X-squares after all of these robots have been around the movement-track at least once and that this evaluation requires that the checker robot go around at most once more. Hence, the robots in T must make two circuits of the movement-track to complete the construction task if it can be completed at all. By the above, as $k \leq |V|$, this requires at most

$$\begin{aligned} 2|V|(2|V| + 16) &= 4|V|^2 + 32|V| \\ &< 25|V|^2 + 400|V| + 1600 \\ &= ((|V| + 8) \times 5)^2 \\ &= |E'|^2 \end{aligned}$$

enabled transition-executions in the worst case, making the construction task in this reduction (c_1, c_2)-completable when $c_1 = 1$ and $c_2 = 2$. To complete the proof, note that in the constructed instance of DesCon^D_{asy}, $|Q| = 3$, $r = |X| = c_1 = 1$, $c_2 = 1$, and $|T| = h = k + 1$. □

Lemma A.5. Dominating set *polynomial-time reduces to* DesCon^D_{asy} *such that in the constructed instance of* DesCon^D_{asy}, $|Q| = 3$, $|E_T| = 12$, $|X| = c_1 = 1$, $c_2 = 2$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Modify the reduction in the proof of Lemma A.3 such that the robots operate asynchronously and apply the (c_1, c_2)-completability argument in the proof of Lemma A.4. To complete the proof, note that in the constructed instance of DesCon^D_{asy}, $|Q| = 3$, $|E_T| = 12$, $|X| = c_1 = c_2 = 1$, and $|T| = h = k + 1$. □

Lemma A.6. Dominating set *polynomial-time reduces to* DesRep^D_{syn} *such that in the constructed instance of* DesRep^D_{syn}, $|Q| = 3$, $r = |X| = |d_X| = c_1 = c_2 = 1$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. In the reduction in the proof of Lemma A.2, modify the checker robot such that the transition that replaces a square of type e_T at p_X when in state q_1 instead replaces a square of type e_{damage}. Observe that this modified checker robot can thus repair damage to X at position p_X. The proof of correctness of this modified reduction is a slight modification of that given in the proof of Lemma A.2. To complete the proof, note that in the constructed instance of DesRep^D_{syn}, $|Q| = 3$, $r = |X| = |d_X| = c_1 = c_2 = 1$, and $|T| = h = k + 1$. □

Lemma A.7. Dominating set *polynomial-time reduces to* DesRep^D_{syn} *such that in the constructed instance of* DesRep^D_{syn}, $|Q| = 3$, $|E_T| = 12$, $|X| = |d_X| = 1$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Apply the reduction modifications in the proof of Lemma A.6 to the reduction in the proof of Lemma A.3. To complete the proof, note that in the constructed instance of DesRep^{D}_{syn}, $|Q| = 3$, $|E_T| = 12$, $|X| = |d_X| = c_1 = c_2 = 1$, and $|T| = h = k + 1$. □

Lemma A.8. Dominating set *polynomial-time reduces to* DesRep^{D}_{asy} *such that in the constructed instance of* DesRep^{D}_{asy}, $|Q| = 3$, $r = |X| = |d_X| = c_1 = 1$, $c_2 = 2$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Apply the reduction modifications in the proof of Lemma A.6 and the completability argument in the proof of Lemma A.4 to the reduction in the proof of Lemma A.2. To complete the proof, note that in the constructed instance of DesRep^{D}_{asy}, $|Q| = 3$, $r = |X| = |d_X| = c_1 = 1$, $c_2 = 2$, and $|T| = h = k + 1$. □

Lemma A.9. Dominating set *polynomial-time reduces to* DesRep^{D}_{asy} *such that in the constructed instance of* DesRep^{D}_{asy}, $|Q| = 3$, $|E_T| = 12$, $|X| = |d_X| = c_1 = 1$, $c_2 = 2$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Apply the reduction modifications in the proof of Lemma A.6 and the completability argument in the proof of Lemma A.4 to the reduction in the proof of Lemma A.3. To complete the proof, note that in the constructed instance of DesRep^{D}_{asy}, $|Q| = 3$, $|E_T| = 12$, $|X| = |d_X| = c_1 = 1$, $c_2 = 2$, and $|T| = h = k + 1$. □

Lemma A.10. Dominating set *polynomial-time reduces to* DesMnt^{D}_{syn} *such that in the constructed instance of* DesMnt^{D}_{syn}, $|Q| = 3$, $r = |X| = |d_X| = c_1 = c_2 = 1$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Observe the following about the reduction in the proof of Lemma A.6: (1) As X has only a single square at p_X, damage to p_X is the only damage that can occur to X; and (2) all vertex neighborhood and checker robots, after being started at E_I, will move forever around the movement-track unless stopped. Hence, any such team that can perform the repair in the instance of DesRep^{D}_{syn} constructed in the proof of Lemma A.6 can also perform the required maintenance in a corresponding instance of DesMnt^{D}_{syn}. To complete the proof, note that in the constructed instance of DesMnt^{D}_{syn}, $|Q| = 3$, $r = |X| = |d_X| = c_1 = c_2 = 1$, and $|T| = h = k + 1$. □

Lemma A.11. Dominating set *polynomial-time reduces to* DesMnt^{D}_{syn} *such that in the constructed instance of* DesMnt^{D}_{syn}, $|Q| = 3$, $|E_T| = 12$, $|X| = |d_X| = c_1 = c_2 = 1$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Apply the argument in the proof of Lemma A.10 to the reduction in the proof of Lemma A.7. To complete the proof, note that in the constructed instance of DesMnt^{D}_{syn}, $|Q| = 3$, $|E_T| = 12$, $|X| = |d_X| = c_1 = c_2 = 1$, and $|T| = h = k + 1$. □

Lemma A.12. Dominating set *polynomial-time reduces to* DesMnt^{D}_{asy} *such that in the constructed instance of* DesMnt^{D}_{asy}, $|Q| = 3$, $r = |X| = |d_X| = c_1 = 1$, $c_2 - 2$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Apply the argument in the proof of Lemma A.10 to the reduction in the proof of Lemma A.8. To complete the proof, note that in the constructed instance of DesMnt^{D}_{asy}, $|Q| = 3$, $r = |X| = |d_X| = c_1 = 1$, $c_2 = 2$, and $|T| = h = k + 1$. □

Lemma A.13. Dominating set *polynomial-time reduces to* DesMnt^{D}_{asy} *such that in the constructed instance of* DesMnt^{D}_{asy}, $|Q| = 3$, $|E_T| = 12$, $|X| = |d_X| = c_1 = 1$, $c_2 = 2$, *and* $|T|$ *and* h *are functions of* k *in the given instance of* Dominating set.

Proof. Apply the argument in the proof of Lemma A.10 to the reduction in the proof of Lemma A.9. To complete the proof, note that in the constructed instance of DesMnt^D_{asy}, $|Q| = 3$, $|E_T| = 12$, $|X| = |d_X| = c_1 = 1$, $c_2 = 2$, and $|T| = h = k + 1$. □

Result B. If either DesCon_{syn} or DesCon_{asy} is polynomial-time tractable when teams are heterogeneous, then $P = NP$.

Proof. The *NP*-hardness of DesCon^D_{syn} and DesCon^D_{asy} follows from the *NP*-hardness of Dominating set and the reductions in Lemmas A.2 and A.4, respectively, in which $h > 1$ in the instances constructed by these reductions. The result then follows from Lemma A.1. □

Result C. If either DesRep_{syn} or DesRep_{asy} is polynomial-time tractable when teams are heterogeneous, then $P = NP$.

Proof. The *NP*-hardness of DesRep^D_{syn} and DesRep^D_{asy} follows from the *NP*-hardness of Dominating set and the reductions in Lemmas A.6 and A.8, respectively, in which $h > 1$ in the instances constructed by these reductions. The result then follows from Lemma A.1. □

Result D. If either DesMnt_{syn} or DesMnt_{asy} is polynomial-time tractable when teams are heterogeneous, then $P = NP$.

Proof. The *NP*-hardness of DesMnt^D_{syn} and DesMnt^D_{asy} follows from the *NP*-hardness of Dominating set and the reductions in Lemmas A.10 and A.12, respectively, in which $h > 1$ in the instances constructed by these reductions. The result then follows from Lemma A.1. □

Result E. If either of $\langle |T|, h, |Q|, r, |X|, |d_X| \rangle$-$\text{DesCon}_{syn}$, -$\text{DesCon}_{asy}$, -$\text{DesRep}_{syn}$, -$\text{DesRep}_{asy}$, -$\text{DesMnt}_{syn}$, or -$\text{DesMnt}_{asy}$ is fp-tractable when teams are heterogeneous, then $FPT = W[1]$.

Proof. Follows from the $W[2]$-hardness of k-Dominating set, the inclusion of $W[1]$ in $W[2]$, the reductions in Lemmas A.2, A.4, A.6, A.8, A.10, and A.12 in which $h > 1$ in the instances constructed by these reductions, and the definition of *FPT*. □

Result F. If either of $\langle |T|, h, |Q|, |E_T|, |X|, |d_X| \rangle$-$\text{DesCon}_{syn}$, -$\text{DesCon}_{asy}$, -$\text{DesRep}_{syn}$, -$\text{DesRep}_{asy}$, -$\text{DesMnt}_{syn}$, or -$\text{DesMnt}_{asy}$ is fp-tractable when teams are heterogeneous, then $FPT = W[1]$.

Proof. Follows from the $W[2]$-hardness of k-Dominating set, the inclusion of $W[1]$ in $W[2]$, the reductions in Lemmas A.3, A.5, A.7, A.9, A.11, and A.13 in which $h > 1$ in the instances constructed in these reductions, and the definition of *FPT*. □

ACKNOWLEDGMENTS

The author would like to thank the anonymous referees for comments and suggestions that helped to significantly improve the technical content and presentation of this article, and Andrew Vardy for very helpful discussions on both autonomous robotics in general and robot team design problems for construction, repair, and maintenance in particular.

REFERENCES

[1] Scott Aaronson. 2005. Complexity theory column 46: *NP*-complete problems and physical reality. *ACM SIGACT News* 36, 1 (2005), 30–52.

[2] Len Adleman, Qi Cheng, Ashish Goel, Ming-Deh Huang, David Kempe, Pablo De Espanes, and Paul Rothemund. 2002. Combinatorial optimization problems in self-assembly. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*. 23–32.

[3] Hadi Ardiny, Stefan Witwicki, and Francesco Mondada. 2015. Are autonomous mobile robots able to take over construction? A review. *Int. J. Rob. Theory Appl.* 4, 3 (2015), 10–21.
[4] Carlos Balaguer and Mohamed Abderrahim. 2008. Trends in robotics and automation in construction. In *Robotics and Automation in Construction*, Carlos Balaguer and Mohamed Abderrahim (Eds.). InTech, 1–20.
[5] Levent Bayındır. 2016. A review of swarm robotics tasks. *Neurocomput.* 172, 8 January (2016), 292–321.
[6] Mauro Birattari, Brian Delhaisse, Gianpiero Francesca, and Yvon Kerdoncuff. 2016. Observing the effects of overdesign in the automatic design of control software for robot swarms. In *Proceedings of ANTS 2016 (Lecture Notes in Computer Science)*, Vol. 9882. Springer, 149–160.
[7] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.
[8] Eric Bonabeau, Sylvain Guérin, Dominique Snyers, Pascale Kuntz, and Guy Theraulaz. 2000. Three-dimensional architectures grown by simple "stigmergic" agents. *Biosyst.* 56, 1 (2000), 13–32.
[9] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. 2013. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intell.* 7, 1 (2013), 1–41.
[10] Rodney A. Brooks. 1992. Artificial life and real robots. In *Towards a Practice of Autonomous Systems: Proceedings of the 1st European Conference on Artificial Life*, Francisco J. Varella and Paul Bourgine (Eds.). MIT Press, 3–10.
[11] Marek Cygan, Fedor V Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer.
[12] Erik Demaine, Mohammad Hajiaghayi, and Dániel Marx. 2014. Minimizing movement: Fixed-parameter tractability. *ACM Trans. Alg.* 11, 2 (2014), 1–29.
[13] Rod Downey and Michael R. Fellows. 2013. *Fundamentals of Parameterized Complexity*. Springer, Berlin.
[14] Paul E. Dunne, Michael Laurence, and Michael Wooldridge. 2003. Complexity results for agent design. *Ann. Math., Comput. Teleinformatics* 1, 1 (2003), 19–36.
[15] Henning Fernau, Torben Hagerup, Naomi Nishimura, Prabhakar Ragde, and Klaus Reinhardt. 2003. On the parameterized complexity of the generalized Rush Hour puzzle. In *Proceedings of the 15th Canadian Conference on Computational Geometry*. 6–9.
[16] Lance Fortnow. 2009. The status of the P versus NP problem. *Commun. ACM* 52, 9 (2009), 78–86.
[17] Gianpiero Francesa and Mauro Birattari. 2016. Automatic design of robot swarms: Achievements and challenges. *Front. Rob. AI* 3, 29 (2016), 1–8.
[18] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability*. W.H. Freeman.
[19] Victor Gerling and Sebastian Von Mammen. 2016. Robotics for self-organised construction. In *IEEE International Worksop on Foundations and Applications of Self* Systems*. IEEE, 162–167.
[20] Alexander Grushin and James A. Reggia. 2008. Automated design of distributed control rules for the self-assembly of prespecified artificial structures. *Rob. Auton. Syst.* 56, 4 (2008), 334–359.
[21] Heiko Hamann, Yara Khaluf, Jean Botev, M. Divband Soorati, Eliseo Ferrante, Oliver Kosak, Jean-Marc Montanier, Sanaz Mostaghim, Richard Redpath, Jon Timmis, et al. 2016. Hybrid societies: Challenges and perspectives in the design of collective behavior in self-organizing systems. *Front. Rob. AI* 3, 11 April (2016), 1–8.
[22] John E. Hopcroft, Jacob Theodore Schwartz, and Micha Sharir. 1984. On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the "warehouseman's problem". *Int. J. Rob. Res.* 3, 4 (1984), 76–88.
[23] Nick Jakobi. 1997. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adapt. Behav.* 6, 2 (1997), 325–368.
[24] Nick Jakobi, Phil Husbands, and Inman Harvey. 1995. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life*, F. Morán (Ed.). Lecture Notes in Computer Science, Vol. 929. Springer, 704–720.
[25] Narendra Karmarkar. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica* 4, 4 (1984), 373–395.
[26] Andreas Kolling, Phillip Walker, Nilanjan Chakraborty, Katia Sycara, and Michael Lewis. 2016. Human interaction with robot swarms: A survey. *IEEE Trans. Hum.-Mach. Syst.* 46, 1 (2016), 9–26.
[27] Christian Komusiewicz and Rolf Niedermeier. 2012. New races in parameterized algorithmics. In *International Symposium on Mathematical Foundations of Computer Science (Lecture Notes in Computer Science)*, Branislav Rovan, Vladimiro Sassone, and Peter Widmayer (Eds.), Vol. 7464. Springer, 19–30.
[28] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. 2013. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Trans. Evol. Comput.* 17, 1 (2013), 122–145.
[29] Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. 2011. Construction of cubic structures with quadrotor teams. In *Robotics: Science & Systems VII*, Hugh F. Durrant-Whyte, Nicholas Roy, and Pieter Abbeel (Eds.). MIT Press, 177–184.
[30] Rolf Niedermeier. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.
[31] Lynne E. Parker and John V. Draper. 1998. Robotics applications in maintenance and repair. In *Handbook of Industrial Robotics (2nd ed.)*, S. Nof (Ed.). Wiley, 1023–1036.

[32] Eric S. Ristad. 1993. *The Language Complexity Game*. MIT Press.
[33] Kamel S. Saidi, Thomas Bock, and Christos Georgoulas. 2016. Robotics in construction. In *Handbook of Robotics*. Springer, 1493–1520.
[34] Touraj Soleymani, Vito Trianni, Michael Bonani, Francesco Mondada, and Marco Dorigo. 2015. Bio-inspired construction with mobile robots and compliant pockets. *Rob. Auton. Syst.* 74, December (2015), 340–350.
[35] Ulrike Stege. 2012. The impact of parameterized complexity to interdisciplinary problem solving. In *The Multivariate Algorithmic Revolution and Beyond*. Number 7370 in Lecture Notes in Computer Science. Springer, Berlin, 56–68.
[36] Ian A. Stewart. 2003. The complexity of achievement and maintenance problems in agent-based systems. *Artif. Intell.* 2, 146 (2003), 175–191.
[37] Ashley Stroupe, Avi Okon, Matthew Robinson, Terry Huntsberger, Hrand Aghazarian, and Eric Baumgartner. 2006. Sustainable cooperative robotic technologies for human and robotic outpost infrastructure construction and maintenance. *Auton. Rob.* 20, 2 (2006), 113–123.
[38] Guy Theraulaz and Eric Bonabeau. 1995. Coordination in distributed building. *Sci.* 269, 5224 (1995), 686.
[39] Guy Theraulaz, Jacques Gautrais, Scott Camazine, and Jean-Louis Deneubourg. 2003. The formation of spatial patterns in social insects: From simple behaviours to complex structures. *Philos. Trans. R. Soc. London, Ser. A* 361, 1807 (2003), 1263–1282.
[40] Mesam Timmar. 2018. *The Computational Complexity of Controller-Environment Co-design Using Library Selection for Distributed Construction*. M.Sc. thesis, Memorial University of Newfoundland.
[41] Mesam Timmar and Todd Wareham. 2019. The computational complexity of controller-environment co-design using library selection for distributed construction. In *Distributed Autonomous Robotic Systems: The 14th International Symposium (Springer Proceedings in Advanced Robotics)*, N. Correll, M. Schwager, and M. Otte (Eds.), Vol. 9. Springer Nature Switzerland AG, 51–63.
[42] Iris van Rooij and Todd Wareham. 2008. Parameterized complexity in cognitive modeling: Foundations, applications, and opportunities. *Comput. J.* 51, 3 (2008), 385–404.
[43] Sebastian Von Mammen, Christian Jacob, and Gabriella Kókai. 2005. Evolving swarms that build 3D structures. In *2005 IEEE Congress on Evolutionary Computation*, Vol. 2. IEEE, 1434–1441.
[44] Todd Wareham. 1999. *Systematic Parameterized Complexity Analysis in Computational Phonology*. Ph.D. Dissertation. University of Victoria.
[45] Todd Wareham. 2015. Exploring algorithmic options for the efficient design and reconfiguration of reactive robot swarms. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communication Technologies*. ICST, Brussels, 295–302.
[46] Todd Wareham, Johan Kwisthout, Pim Haselager, and Iris van Rooij. 2011. Ignorance is bliss: A complexity perspective on adapting reactive architectures. In *Proceedings of the 1st Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics*, Vol. 2. 1–5.
[47] Todd Wareham and Andrew Vardy. 2018. Putting it together: The computational complexity of designing robot controllers and environments for distributed construction. *Swarm Intell.* 12, 2 (2018), 111–128.
[48] Todd Wareham and Andrew Vardy. 2018. Viable algorithmic options for designing reactive robot swarms. *ACM Trans. Auton. Adapt. Syst.* 13, 1 (2018), 5:1–5:22.
[49] Justin Werfel and Radhika Nagpal. 2008. Three-dimensional construction with mobile robots and modular blocks. *Int. J. Rob. Res.* 27, 3–4 (2008), 463–479.
[50] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. 2014. Designing collective behavior in a termite-inspired robot construction team. *Sci.* 343, 6172 (2014), 754–758.
[51] Stefan Wismer, Gregory Hitz, Michael Bonani, Alexey Gribovskiy, and Stéphane Magnenat. 2012. Autonomous construction of a roofed structure: Synthesizing planning and stigmergy on a mobile robot. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 5436–5437.
[52] Michael Wooldridge and Paul E. Dunne. 2002. The computational complexity of agent verification. In *Intelligent Agents VIII*. Springer, 115–127.
[53] Seung-kook Yun, Mac Schwager, and Daniela Rus. 2011. Coordinating construction of truss structures using distributed equal-mass partitioning. In *Robotics Research*, C. Pradalier, R. Siegwart, and G. Hirzinger (Eds.). STAR, Vol. 70. Springer, Berlin, 607–623.

Received September 2018; revised March 2019; accepted May 2019