Swarm Control for Distributed Construction: A Computational Complexity Perspective

TODD WAREHAM, Memorial University of Newfoundland, Canada
RONALD DE HAAN, Universiteit van Amsterdam, The Netherlands
ANDREW VARDY, Memorial University of Newfoundland, Canada
IRIS VAN ROOIJ, Radboud Universiteit, The Netherlands

Over the last 20 years, human interaction with robot swarms has been investigated as a means to mitigate problems associated with the control and coordination of such swarms by either human teleoperation or completely autonomous swarms. Ongoing research seeks to characterize those situations in which such interaction is both viable and preferable. In this paper, we contribute to this effort by giving the first computational complexity analyses of problems associated with algorithm, environmental influence, and leader selection methods for the control of robot swarms performing distributed construction tasks. These analyses are done relative to a simple model in which swarms of deterministic finite-state robots operate in a synchronous error-free manner in 2D grid-based environments. We show that all three of our problems are polynomial-time intractable in general and remain intractable under a number of plausible restrictions (both individually and in many combinations) on robot controllers, environments, target structures, and sequences of swarm control commands. We also give the first restrictions relative to which these problems are tractable, as well as discussions of the implications of our results for both the design and deployment of swarm control assistance software tools and the human control of robot swarms.

CCS Concepts: • **Human-centered computing → Systems and tools for interaction design**; • **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: human-robot interaction, swarm robotics, construction, computational complexity

ACM Reference Format:
Todd Wareham, Ronald de Haan, Andrew Vardy, and Iris van Rooij. 2020. Swarm Control for Distributed Construction: A Computational Complexity Perspective. *ACM Trans. Hum.-Robot Interact.* 0, 0, Article 0 (2020), 45 pages. https://doi.org/0

1 INTRODUCTION

Much work has been done over the last 40 years on multi-robot systems that can perform tasks more effectively, efficiently, and robustly than single robots. A central issue in such systems is who or what is responsible for control and co-ordination of the robots in a system. In Sheridan and

Authors' addresses: Todd Wareham, harold@mun.ca, Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada, A1B 3X5; Ronald de Haan, me@ronalddehaan.eu, Institute for Logic, Language, and Computation, Universiteit van Amsterdam, Amsterdam, The Netherlands; Andrew Vardy, av@mun.ca, Department of Computer Engineering and Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada, A1B 3X5; Iris van Rooij, iris.vanrooij@donders.ru.nl, Donders Institute for Cognition, Radboud Universiteit, Nijmegen, The Netherlands.

2573-9522/2020/0-ART0 $15.00
https://doi.org/0

Verplank's 10-point Levels of Automation (LOA) scale [44], options for control and co-ordination fall on a spectrum between systems with fully autonomous members, i.e., robots swarms (LOA level 10) and teleoperated systems where humans control everything (LOA level 1). Quite aside from physical implementation problems related to issues like robot hardware and human-robot communication, both of these extremes on the spectrum have proven inherent computational difficulties (see [50, 59, 62, 63] and [8, 24], respectively). Hence, intermediate levels of control and co-ordination are of great interest.

One popular option is human-swarm hybrid systems, in which robots and human operators share system control and co-ordination. The general mode of operation and benefits of such systems are succinctly summarized on page 9 of [28]:

> For the most part, swarms are expected to operate autonomously. However, the presence of a human operator can be beneficial and even necessary since the operator could: 1) recognize and mitigate shortcomings of the autonomy; 2) have available "out-of-band" information not accessible to the autonomy and that can be utilized to increase performance; and 3) convey changes in intent as mission goals change.

Such hybrid systems fall on LOA level 7 or higher [28, page 19]. Human operators can exert intermittent control over swarms or individual swarm members by various methods. Four types of methods have been investigated to date [28, Section III-D]:

(1) **Parameter Setting**: Change controller-algorithm parameters of all or selected swarm members;
(2) **Algorithm Selection**: Change the controller algorithms of all or selected swarm members to new controller algorithms selected from a provided controller algorithm library;
(3) **Environmental Influence Selection**: Alter the behaviour of the swarm by placing guiding marks, pheromone sources, and/or beacons either virtually or physically in the environment; and
(4) **Leader Selection**: Designate selected swarm members as leaders and either transiently or persistently override the controllers of these leaders to alter the behaviour of the swarm as a whole.

As there does not appear to one universally applicable swarm control type, the following is necessary [28, page 22]:

> It is essential to determine which general [swarm control] types and their various implementations are suitable for which kinds of tasks, environments, communication and timing constraints, and other swarm-specific circumstances. In addition, when multiple types are suitable, one should attempt to compare effectiveness, scope, and impact of these control type, i.e., how many robots they affect directly and indirectly.

To date, this has been addressed by three strands of research (see [28, page 20] and references). **[R1(C6)] The first of these strands is theoretical models and analyses, which either determine feasible and optimal control inputs or (as in the case of Lewis' cognitive complexity model [30]) broadly characterize types of human-robot interaction situations. The second involves simulations in cases where theoretical models and analyses are not applicable. Finally, there are user studies and prototype systems (the latter possibly incorporating swarm control assistance software tools for human operators) in simulation or with real robots, which are used to address more complex scenarios or verify theoretical insights**.

A valuable complement to this work would be to establish the general algorithmic options for the various types of swarm control. This can be done using the tools and techniques of computational complexity analysis [11, 18]. **[R2(C2a),R2(C5)] For a given computational problem, such analyses determine whether or not there is an efficient algorithm for that problem relative to some criterion of efficiency, i.e., worst-case runtime or memory usage. These problems may be simplifications or generalizations of actual problems of interest; the former is done to make analysis easier and the latter (via collected analyses of restricted versions of the generalized problem) to map out those restrictions that do and do not allow efficient solvability, i.e., to determine the frontier of tractability for the problem [18, Section 4.1].**[1] The results of such analyses can be used not only to establish those situations in which known algorithms are the best possible but also to guide the development of new algorithms (by highlighting relative to which restrictions and types of efficient solvability such algorithms can and cannot exist). **In the case of swarm control, this would be** useful not only in creating the best possible swarm control assistance software tools for human operators but also (using the framework described in [54]) providing more detailed and comprehensive models of human operator cognitive effort to better guide the design and implementation of human-robot systems.

1.1 Previous Work

The human computational effort in human-robot interaction has been investigated using the notion of cognitive complexity developed in [30, 31] (see also [28, Section III-A]). This scheme **[R1(C7)] adapts** the $O(f(n))$ ("Big-Oh") asymptotic worst case time complexity notation from computational complexity analysis to **[R1(C7)] describe** three categories of tasks in terms of human operator effort in swarm control[2] [28, page 13]:

(1) $O(1)$ (e.g., swarm flocking or rendezvous), in which a group of robots are all performing a single task autonomously such that the group can be treated as a single entity and one (or a fixed number of) human operator(s) can control any number of robots;
(2) $O(n)$ (e.g., search and rescue), in which a group of robots are performing independent tasks autonomously such that each of the robots in the group requires the same amount of operator interaction and the total operator effort scales linearly with the number of robots in the group; and
(3) $O(>n)$ (e.g., box pushing), in which a group of robots must co-ordinate to perform a common task such that dependencies between robots create cascading cognitive demands on human operators as the number of robots grows.

These categories essentially characterize both the degree to which co-ordination among swarm members can be automated and whether or not a human-in-the-loop system can be scaled to larger numbers of robots [30, page 139]. Swarm control situations requiring $O(1)$ cognitive effort are considered the most desirable, and guidelines for designing and dealing effectively with all three

[1] **These restrictions can be either numerical, e.g., the robot swarm can have at most n members, or structural, e.g., each robot can only move such that it can always perceive at least one other robot in the swarm. In the case of numerical restrictions, the goal is to precisely delineate the value of the problem aspect being restricted at which tractability becomes intractability. A classic example of such a sharply defined frontier of tractability is for the problem** CNF-SAT **(Does a given propositional logic formula in conjunctive normal form, i.e., a formula in which clauses of OR-ed variables are AND-ed together, have an assignment of truth values to its variables which makes that formula true?) relative to maximum clause length l. If $l = 2$ CNF-SAT is polynomial-time solvable but if $l = 3$ it is not [18, Problem L01].**

[2] **[R1(C7,C8),R4(C2)] Note that this notation is not to be understood as denoting "number of elementary operations as a function $f()$ of input size n", which would be the traditional computer science interpretation used in computational complexity analyses such as those described in the third paragraph of this subsection.**

categories of systems are given on pages 162–163 of [30]. Though useful, this cognitive complexity scheme is incomplete — this is so because, as noted on pages 13–14 of [28], the primary purpose of this scheme is to emphasize the attention effort of the human operator in controlling a multi-robot system, and other factors (e.g., human-robot communication, swarm state estimation and prediction, design of swarm control inputs) may increase the overall required cognitive effort of swarm control greatly.

[R4(C5)] There are many tasks relative to which swarm robotics and control can be investigated [4, 5]. Given that some of the most impressive activities performed by insect swarms involve creating structures, e.g., wasp and termite nests, it is not surprising that distributed construction has been of key interest. From the start of such research over 25 years ago [48], the focus has by large been on construction by fully autonomous robot swarms (Collective Robotic Construction (CRC)), e.g., [2, 19, 39, 45, 47, 67]. Research on CRC has in the past been restricted to academia, in large part because of the conservatism of the real-world construction industry and its concern with safety issues when robots and humans work in shared environments [3, 43]. Recent research has however focused much more on Collective Human-Robot Construction (CHRC) [20, 49, 51]. CHRC is seen as both an important research challenge in CRC [39, Page 5] as well as a critical technology for fully integrating robots into real-world construction. Though many researchers see the role of robots in CHRC primarily as assistants to human beings in structure fabrication [39, Page 10], there are a growing number who foresee opportunities for humans and robots to collaborate in designing structures as well [20, 49].

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 [1, 8, 12, 13, 24, 46, 69]. **Seven** complexity-theoretic papers to date incorporate both autonomous robots and a suitably simple and explicit model of robot architecture and environment that allows investigation of possible restrictions that could yield tractability [50, 59–64]. Three of these [59, 61, 63] consider navigation tasks performed by robots with deterministic Brooks-style subsumption [59, 61] and single-state finite-state [63] reactive controllers, respectively. The other **four** consider construction-related tasks performed by robots with deterministic finite-state controllers relative to robot controller and environment design in a given environment where robot controllers are designed from scratch [62, 64], robot swarm design where swarm members are designed by selection from a provided library [60, 64], and robot swarm / environment co-design where swarm members are designed by selection from a provided library [50]. No complexity-theoretic analysis to date has considered the computational difficulty associated with swarm control.

1.2 Summary of Results

In this paper, we present the first computational complexity analyses of **[R2(C5)] problems that are generalizations of** the algorithm, environmental influence, and leader selection swarm control methods described above relative to the distributed construction task [39]. These problems can be stated informally as follows (and are described in more detail in **Sections 2.2 and 2.3**):

- Robot Algorithm Selection (SelAlg): Derive (if one exists) a set of changes to the algorithms of the robots in a given swarm T relative to a given robot-algorithm library that occur over a specified period of time and enable the robots in T to create a specified structure at a specified location in a given environment.
- Environmental Influence Selection (SelEnvInf): Derive (if one exists) a set of changes to a given environment E that occur over a specified period of time and enable the robots in a given swarm to create a specified structure at a specified location in E.

Leader Selection **[R4(C1)] (SelLead)**: Derive (if one exists) a set of changes to the positions of the robots in a given swarm T that occur over a specified period of time and enable the robots in T to create a specified structure at a specified location in a given environment.

We consider the following types of efficient solvability (described in more detail in Section 3):

(1) Polynomial-time exact solvability, such that a polynomial-time algorithm produces the correct output for a given input either (a) all the time [18] or (b) when such an output is known to exist (promise solvability).
(2) Polynomial-time approximate solvability, such that a polynomial-time algorithm produces the correct output for a given input either (a) in all but a small number of cases [22] or (b) with a high probability [35].
(3) Effectively polynomial-time exact restricted solvability (fixed-parameter (fp-)[11] and $|x|^k$-tractability), such that an algorithm produces the correct output for a given input in what is effectively polynomial time when certain aspects of that input are of restricted value, e.g., the number of robots in the swarm or the total number of swarm control inputs issued by the human operator is small.

Our analyses are done in terms of the distributed construction model defined in [62] in which robots equipped with finite-state controllers move in a synchronous non-continuous manner in grid-based 2D environments and swarms are restricted to complete their construction tasks in a polynomial number of timesteps. Relative to the types of solvability listed above and various conjectures that are widely believed to be true within the Computer Science community, e.g. $P \neq NP$ [16, 18], we prove the following results (Section 3):

- Though SelAlg is solvable in sense (1a) above when all robots in the swarm T have the same algorithm **[R4(C4)] (i.e., the swarm is homogeneous)** and all robot-algorithm changes occur prior to the first timestep (Result A.1), this is not the case for SelAlg when the robots in the swarm can each have one of two algorithms **[R4(C4)] (i.e., the swarm is heterogeneous)** or for either SelEnvInf or SelLead when **[R4(C4)] swarms are homogeneous** and all environment- and position-changes occur prior to the first timestep (Result A.2).
- **[R4(C4)] Neither SelAlg when swarms are heterogeneous and all robot-algorithm changes occur prior to the first timestep nor SelEnvInf or SelLead when swarms are homogeneous and all environment- and position-changes occur prior to the first timestep is solvable in sense (1b) above (Result A.3).**
- **Neither SelAlg when swarms are heterogeneous and all robot-algorithm changes occur prior to the first timestep nor SelEnvInf or SelLead when swarms are homogeneous and all environment- and position-changes occur prior to the first timestep is solvable in sense (2a) or (2b) above (Results A.4 and A.5).**
- Neither of SelAlg, SelEnvInf, or SelLead is fixed-parameter or $|x|^k$ tractable in sense (3) above relative to restrictions on a number of aspects of these problems related to robot swarm size and structure, the algorithms used by individual robots, environment size and structure, and the degree of swarm control (see Table 1 for the full list of aspects considered here). This fixed-parameter and $|x|^k$ intractability holds when many combinations of these aspects **[R4(C4)] are simultaneously restricted**, often to small constant values (Results B.SA.1–B.SA.7 and C.SA.1–C.SA.5 (SelAlg), Results B.SE.1–B.SE.4 and C.SE.1 and C.SE.2 (SelEnvInf), and Results B.SL.1–B.SL.4 and C.SL.1–C.SL.3 (SelLead)) (see also Tables 2–5). That being said, there are several combinations of aspect-restrictions that do yield fixed-parameter and $|x|^k$ tractability (Results B.SA.8, B.SA.9, and C.SA.6 (SelAlg), Results B.SE.5 and C.SE.3 (SelEnvInf), and Results B.SL.5 and C.SL.4 (SelLead)), though neither of SelAlg, SelEnvInf, nor SelLead have a particularly desirable sublinear form of $|x|^k$ tractability relative to the maximum

allowable number of algorithm-, environment-, or leader-changes (Results C.SA.7, C.SE.4, and C.SL.5).

As is discussed in Section 4.1, our use of simplified models as well as our problem- rather than algorithm-oriented method of analysis has exposed what we believe to be under-appreciated underpinnings of the computational difficulty of swarm control arising from the combinatorial choices in swarm control inputs. The implications of our results for swarm control assistance software tools and the human control of swarms are discussed in Sections 4.3 and 4.4, respectively; as part of the latter, we also describe how computational complexity analyses such as ours may allow a detailed and comprehensive assessment of the human cognitive effort associated with swarm control.

Two issues with respect to the results listed above and their derivation should be noted. First, to aid readability, various definitions and lemmas stated in previous papers are repeated here. **[R2(C7)] In the interests of concision and avoiding self-plagiarism, abbreviated forms are often used (most notably, in Section 2.1, the "short form" of definitions associated with our distributed construction model as given in [50]). [R2(C1)] As some proofs of results presented here are related to proofs given previously in [60, 62], there are potential concerns about the originality of results presented in the current paper. These relationships can be categorized as follows:**

1. **As is described in more detail in Section 2, problems DesCon and EnvDes investigated previously in [60] and [62], respectively, are special cases of problems SelAlg and SelEnvInf investigated here. Hence, several lemmas used to prove intractability results in this paper for SelAlg and SelEnvInf (namely, Lemmas A.5, A.6, and A.13–A.16) are straightforward adaptations of lemmas used to prove intractability results for DesCon and EnvDes in [60, 62]. This is indicated by the initial phrase "Adapted from" at the beginning of the proofs of these lemmas in the appendix.**
2. **Two lemmas used to prove intractability results in this paper for SelAlg and SelLead (namely, Lemmas A.9 and A.20) build on lemmas in [60, 62] by invoking the ability of problems SelAlg and SelLead in the current paper to allow swarm control changes to occur over a period of time. These are indicated by the phrase "Inspired by" at the beginning of the proofs of these lemmas in the appendix.**
3. **All other lemmas that are used to prove intractability results in this paper (namely, Lemmas A.7, A.8, A.10–12, and A.17–19) are new.**

The results derived relative to the proofs in the first category (parts of Results A.2, A.4, and A.5 and all of Results A.1, B.SA.1, B.SA.2, B.SE.1–4, C.SE.1, C.SE.2, and C.SA.7) are outnumbered two to one by the results derived relative to the second (Results B.SA.4, B.SL.4, C.SA.2, and C.SL.3) and third (parts of Results A.2, A.4, and A.5 and all of Results X.1, A.3, B.SA.3, B.SA.5–9, B.SE.5, B.SL.1–3, B.SL.5, C.SA.1–6, B.SE.3, C.SL.1–4, C.SE.4. and C.SE.5) categories. Hence, though some of the results presented here resemble those presented previously in [60, 62], the majority of the results in the current paper are in fact new.

1.3 Organization of Paper

This paper is organized as follows. In Section 2, we describe our robot controller, environment, target structure, and swarm control models and use these to formalize the swarm control problems for distributed construction that we will analyze. In Section 3, we consider viable algorithmic options for these problems relative to several popular types of exact unrestricted, approximate unrestricted, and exact restricted efficient solvability. All proofs of stated results are given in the appendix. In Section 4, we discuss a variety of issues raised by our results, including the completeness and

generality of these results (**Sections 4.1 and 4.2, respectively**) and the implications of these results for both swarm control assistance software tools (Section 4.3) and human control of swarms in general (Section 4.4); the last of these also describes how computational complexity analysis (and in particular parameterized complexity analysis) can be used to create a detailed and comprehensive assessment of human cognitive effort for swarm control. Finally, our conclusions and directions for future work are given in Section 5.

2 FORMALIZING SWARM CONTROL FOR DISTRIBUTED CONSTRUCTION

In this section, we first review the basic entities in the model of structure creation by robot swarms given in [60, 62] (Section 2.1). We then describe the three types of swarm control from [28] and the generalizations of these types that will be of interest to us (Section 2.2). Finally, we formalize the computational problems associated with these three types of swarm control relative to the distributed construction task that we will analyze in the remainder of the paper (Section 2.3).

2.1 Robot Swarms for Distributed Construction

[R2(C7)] We here review the basic entities in our model of distributed construction by robot swarms — namely, environments, target structures, individual robots, and robot swarms. In the interests of concision and avoiding self-plagiarism, this review is the "short form" originally given in [50]; readers wishing more details and examples should consult [60, 62].

The basic entities in our model are as follows:

- **Environments and Target Structures**: Our robots operate in a finite 2D square-based environment E in which each square is either a freespace (which a robot can occupy or move through) or an obstacle and has a square-type, e.g., grass, gravel, wall, drawn from a set E_T. 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 southwest-most corner of E; we also assume an arbitrary ordering over the elements in E_T. A structure X in an environment E is a two-dimensional pattern of squares of special square-type $e_X \in E_T$ in an $m \times n$ grid whose location in E is specified relative to the position p_X of southwest-most corner of the structure-grid in the environment-grid.
- **Robots**: Each robot occupies a square in E 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. Each robot has a sensing-distance bound r such that the robot can sense the type of the square at any position within Manhattan distance $r \geq 0$ of the robot's current position (with $r = 0$ corresponding to the square on which the robot is standing).[3] 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 occupying 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 one of the robot's current position to type e via predicates of the form $enmod(e, pos)$ where pos is specified as for $enval()$.
  Each robot has a finite-state controller and is hence known as a Finite-State Robot (FSR). Each such controller consists of a set Q of states linked by transitions, where each transition

[3] This set of sensed squares is the von Neumann neighbourhood of range r centered on the robot's position and consists of $1 + 2r(r + 1) = 2r^2 + 2r + 1$ squares [65].

(q, f, x, dir, q') between states q and q' has a propositional logic trigger-formula f, an environment modification specification x, and a movement-direction $dir \in \{goNorth, goSouth, goEast, goWest, stay\}$. **[R1(C9), R2(C7), R4(C3)] Trigger-formulas and modification specifications are typically stated in terms of predicates** $enval()$ **and** $enmod()$**, respectively. Examples of such formulas and specifications relative to a robot** r **are** $(enval(e_{robot}, (-1, 0))$ and $enval(e_{grass}, (0, 0)))$ or not $enval(e_{robot}, (0, -3))$ **(is it the case that either** r **is on a grassy square and has another robot in the square immediately next to it to the East or there is no robot three squares South of** r**?) and** $enmod(e_X, (0, 1))$ **(place a block of construction material in the square to the immediate North of** r**). Both of these specifications can also be stated as a special symbol** $*$**, which is interpreted as follows**: If $f \neq x \neq *$ and the transition's trigger-formula evaluates to $True$, i.e., the transition is enabled, this causes the environment-modification specified by x to occur, the robot to move one square in direction dir, and the robot's state to change from q to q'. If $f = *$, the transition executes if no other transition executes (making this in effect the default transition); if $x = *$, no environment-modification is made.

- **Robot swarms**: A swarm T consists of a set of the robots described above, where there may be more than one robot with the same controller in a swarm; as such, we allow both homogeneous and heterogeneous swarms. Let T_i denote the ith robot in the swarm. Each square in E can hold at most one member of T; if at any point in the execution of a task two robots in a swarm attempt to occupy or modify the same free space or a robot attempts to occupy the same space as an obstacle, the execution terminates and is considered unsuccessful. A **positioning** of T in E is an assignment of the robots in T to a set of $|T|$ squares in E. **[R3(C1)] Swarm members can move either synchronously or asynchronously as specified; however, in both cases, once movement is triggered, it is atomic in the sense that the specified movement is completed.**
  Note that for simplicity, robots in our swarms do not communicate with each other directly — rather, they communicate with each other indirectly through their sensed presences in and changes they make to the environment, i.e., via extended stigmergy [66]. Though this does not allow private robot-to-robot communication, it does allow non-trivial messages between robots (if the set of available environmental modifications that can be made and perceived by robots is of even moderate size) that does not require temporal synchronization between the robots [39, page 4]. Moreover, such communication by presence and stigmergy is arguably typical of biological and artificial swarms.

Notions of deterministic and time-bounded robot and swarm operation were introduced in [62] to ensure that requested structures are created by swarms reliably and quickly. We use the notion of deterministic robot and swarm operation introduced in [62] as extended in [50] (i.e., requiring that at any time as the swarm operates in an environment, all transitions enabled in a robot relative to the current state of that robot perform the same environment modifications and progress to the same next state).[4]

[4] As noted in Footnote 1 and the main text on pages 115–116 of [62], our operational conception of FSR determinism is very different than the traditional absolute definition of determinism for finite-state automata [23, Section 2.2], which is guaranteed by the structure of the state-transition functions of those automata. Our operational definition is necessary because FSR can sense and enable transitions relative to arbitrary patterns of squares (including the presence of other FSR) within radius r of their current position, and the number of such patterns that can be encountered (especially if r and $|E_T|$ are not of constant value) is exceptionally large. Indeed, for unrestricted r, the problem of assessing whether or not the transitions in a given FSR are operationally deterministic in all possible environments of a specified size and shape (even if there are no other FSR in the environment and the given FSR has only two states and two transitions) is

We also retain the second notion, encapsulated as (c_1, c_2)-**completability**, which requires that each swarm complete its task within $c_1|E|^{c_2}$ timesteps for constants c_1 and c_2. However, we here make two modifications. First, as done in [50], we broaden the timestep bound to $c_1(|E| + |Q|)^{c_2}$, in order to accommodate FSR that make a number of internal state-changes without moving. Second, we fix a technical flaw in our previous papers. In those papers, c_1 and c_2 were part of the inputs to our analyzed problems. **[R1(C2),R3(C2),R4(C6)] Following standard practice in encoding problem inputs [18, Page 10], base-10 decimal values c_1 and c_2 were encoded in base-2 binary, e.g., $c_1 = 7_{dec} = 4+2+1 = 2^2+2^1+2^0 = 111_{bin}$. Note that binary notation is of length logarithmic in the value, e.g., $|111_{bin}| = 3 \approx \log_2 7_{dec}$. This has the unfortunate effect of rendering the task completion bound exponential rather than polynomial in the input size, $c_1(|E| + |Q|)^{c_2} = 2^{\log_2 c_1}(|E| + |Q|)^{2^{\log_2 c_2}}$. One can get around this by assuming that c_1 and c_2 are encoded in the input in base-1 unary, e.g., $7_{dec} = 1111111_{un}$. However, this has the equally unfortunate effect of allowing exponential-time computations to be artificially reduced to polynomial time by "padding" the input length with large unary encoded values [18, Pages 9–10]. To avoid both of these difficulties, we chose here to remove c_1 and c_2 from the problem input and assume that suitable constant values of c_1 and c_2 are specified beforehand. To ensure generous but still low-order polynomial swarm runtime bounds, we will assume that $c_1 = 10$ and $c_2 = 3$.**

[R2(C2a)] Given this model of distributed construction, we say that the task of constructing structure X at position p_X in E is (c_1, c_2)-completable by a swarm T relative to a positioning p_I of T in E if the members of T, when started at p_I, operate as dictated by their control algorithms and succeed in constructing X at p_X within at most $c_1(|E| + |Q|)^{c_2}$ timesteps. In the next subsection, we will similarly formalize how the operation of swarms during this construction process can be modified by the swarm control methods described in Section 1.

2.2 Types of Swarm Control

[R2(C2a)] In this paper, we shall focus on three of the four methods of swarm control described in Section III-D of [28] — namely, algorithm selection, environment influence selection, and leader selection. In the literature, variants of each of these methods have been investigated:

1. **Algorithm selection where either all or selected subsets of the swarm have their control algorithms changed simultaneously.**
2. **Environment influence selection where environmental changes are (a) physical or virtual and (b) persistent or transient (e.g., decaying pheromones).**
3. **Leader selection where selected leaders are (a) explicitly or not explicitly indicated as leaders to other swarm members and (b) always or temporarily under human control after selection.**

Rather than analyze each of these variants separately, we will instead analyze the following generalizations of these methods:[5]

not polynomial-time tractable unless $P = NP$ (see Result X.1 in the appendix). Given the above, an individual FSR is thus not itself deterministic in our sense but rather the operation of that FSR is deterministic in the context of a particular FSR swarm operating in a particular environment.

[5] **[R2(C2b)] It has been pointed out to us that, given an appropriately detailed library of robot controller algorithms, all three of these methods can be simulated by and hence viewed as special cases of, algorithm selection. While this is appealing on the grounds of mathematical simplicity, we have chosen to retain three separate methods to follow the standard method classification laid out in Section III-D of [28] and hence allow more obvious and hopefully easier connection of our results with those derived in the HRI research community.**

(1) Algorithm selection where, at any time during swarm operation, one or more subsets of swarm members may have their control algorithms changed simultaneously, with each subset having a different new control algorithm.
(2) Environmental influence selection where, at any time during swarm operation, environmental changes are physical and persistent.
(3) Leader selection where, at any time during swarm operation, selected leaders are not explicitly indicated as leaders to other swarm members and are only human control for a single timestep after being selected.

Note that these generalized methods have all studied variants as special cases (with transient environmental changes simulated by later environmental changes that undo earlier ones and explicit indication of leaders simulated via leader-made environmental modifications). Hence, by the logic of problem generalization laid out in Section 1, these methods can (under the appropriate restrictions) be used analyze those variants. When combined with additional restrictions (see parameters $t_{\max}$, k_a, k_e, and k_l), these methods will also allow particularly fine-grained analyses of the computational complexity associated with the time-periods in which controller algorithm- and environment-changes take place and the maximum numbers of allowable changes.

Given the above, consider the following formalizations of these three generalized methods of swarm control:

(1) **Algorithm Selection**: In this method of swarm control, some members of the swarm are selected and have their controllers changed to other controllers selected from a given controller-library L. We will assume that all controllers in L are behaviorally distinct. Let $C_A = \{c_1, c_2, \ldots, c_{|C_A|}\}$ be a set of such changes where each change $c_i = (j, k, l)$ specifies the timestep j at which the change takes place and the swarm-member T_k whose controller is to be replaced with L_l from the given controller-library L. In our problems, we will limit the maximum number of allowable changes in C_A using the parameter k_a.
(2) **Environmental Influence**: In this method of swarm control, some number of squares in E are selected and have their types changed to other square-types selected from $E_T - \{e_X\}$. These changes correspond to persistent guiding marks in the physical environment, cf. persistent beacons that modify the controller algorithms of all robots within a specified radius [27, page 91] or transient pheromones in the physical or virtual environment [28, page 17]. Let $C_E = \{c_1, c_2, \ldots, c_{|C_E|}\}$ be a set of such changes where each change $c_i = (j, k, l, m)$ specifies the timestep j at which the change takes place and the environment-square $E_{k,l}$ whose square-type is to be changed with the mth type in E_T. In our problems, we will limit the maximum number of allowable changes in C_E using the parameter k_e.
(3) **Leader Selection**: In this swarm control method, some members of the swarm (called **leaders**) are selected and their control algorithms are temporarily overridden to make these leaders move to new positions. Without loss of generality, we will assume that each of these moves is to one of the four surrounding squares to the north, south, east, or west of the robot's original position. As robots so selected are not distinguishable from the other robots and are influenced intermittently, they correspond to transient tacitly-indicated leaders, cf. persistent explicitly-indicated leaders that are under continuous control by human operators (i.e., teleoperation) [28, pages 18–19]. Let $C_L = \{c_1, c_2, \ldots, c_{|C_L|}\}$ be a set of such changes where each change $c_=(j, k, l, m)$ specifies the timestep j at which the change takes place and the x and y displacements **[R1(C10)]** l **and** $m \in \{-1, 0, 1\}$ by which swarm member T_k is moved. In our problems, we will limit the maximum number of allowable changes in C_L using the parameter k_l.

All changes are assumed to occur at or before timestep $t_{\max}$, where $1 \leq t_{\max} \leq c_1(|E| + |Q|)^{c_2}$ and c_1 and c_2 are the constants used to specify (c_1, c_2)-completability. There may be multiple changes occurring in a particular timestep. **[R2(C2c)] In the cases of algorithm and environmental influence selection, all changes are assumed to occur simultaneously at the beginning of a timestep prior to swarm member environment sensing and action. In the case of leader selection, changes are assumed to occur simultaneously during the swarm member action phase, such that only leaders move with respect to the specified changes and all other swarm members move as specified by their controller algorithms. Given this**, no two changes in a timestep can affect the same entity. If $t_{\max} = 1$, all changes occur in the first timestep and the relative timing of changes is not an issue; however, this is most certainly not the case when $t_{\max} > 1$, as factors like necessary swarm stabilization time after certain operator commands and degradation in swarm member performance between operator commands mean that the same changes under different timings can have radically different and sometimes even adverse effects on swarm behaviour [28, page 19].

2.3 Computational Problems

We can now **[R2(C2a)] fuse the model of distributed construction in Section 2.1 with our formalizations of swarm control methods in Section 2.2 to define the computational problems which will analyze in this paper:**

Robot Algorithm Selection (SelAlg)
Input: An environment E based on square-type set E_T, an FSR swarm T based on controllers from library L, an initial positioning p_I of T in E, a structure X, a position p_X of X in E, and positive integers $t_{\max}$ and $k_a \leq |T| \times t_{\max}$.
Output: A set C_A of k_a algorithm changes to T relative to L and $t_{\max}$ such that the task of constructing X at p_X when T is initially positioned at p_I and the changes in C_A are subsequently applied is (c_1, c_2)-completable, if such a C_A exists, and special symbol $\perp$ otherwise.

Environmental Influence Selection (SelEnvInf)
Input: An environment E based on square-type set E_T, an FSR swarm T, a structure X, an initial positioning p_I of T in E, a position p_X of X in E, and positive integers $t_{\max}$ and $k_e \leq |E| \times t_{\max}$.
Output: A set C_E of k_e environmental changes to E relative to $E_T - \{e_X\}$ and $t_{\max}$ such that the task of constructing X at p_X in E when T is initially positioned at p_I and the changes in C_E are subsequently applied is (c_1, c_2)-completable, if such a C_E exists, and special symbol $\perp$ otherwise.

Robot Leader Selection (SelLead)
Input: An environment E based on square-type set E_T, an FSR swarm T, a structure X, an initial positioning p_I of T in E, a position p_X of X in E, and positive integers $t_{\max}$ and $k_l \leq |T| \times t_{\max}$.
Output: A set C_L of k_l changes to the positions of members of T relative to $t_{\max}$ such that the task of constructing X at p_X in E when T is initially positioned at p_I and the changes in C_L are subsequently applied is (c_1, c_2)-completable, if such a C_L exists, and special symbol $\perp$ otherwise.

Unless stated otherwise, all swarm operation in this paper will be synchronous. Without loss of generality, we will assume that each member of T starts operating in the initial state q_0 of its associated controller, either at the start of the construction task or whenever that controller is changed. Moreover, in the case of problem SelLead, if the execution of a change causes two robots in a swarm to attempt to occupy the same space or an individual robot to occupy the same space as an obstacle then the execution of the task terminates and is considered unsuccessful. Note that problems SelAlg and SelEnvInf when $t_{\max} = 1$ (i.e., all algorithm- and environment-changes are

done prior to the first timestep) are for all **[R4(C1)]intents and purposes** problems DesCon and EnvDes investigated previously in [60] and [62], respectively. This similarity will allow us to easily adapt several previously-published result proofs for DesCon and EnvDes to derive analogous results for SelAlg and SelEnvInf (most notably, Results B.SA.1, B.SA.2, and B.SE.1–B.SE.4) in Section 3 (see Section 1.2 for further discussion of this issue).

At this point, the reader may be concerned that the problems we analyze are overly abstract, both in the synchronous non-continuous manner in which our robots operate in 2D grid-based environments and their ignoring of known issues in the implementation of swarm control such as communication latency and bandwidth between human operators and swarm members [28, Section III-B] and swarm state estimation and prediction by human operators in the face of incomplete observations [28, Section III-C] (courtesy of our assumptions of complete swarm observability, deterministic swarm operation, and (c_1, c_2)-completability of tasks). We do not deny that **[R1(C11)] these** issues are critical to real-world swarm control. However, as we shall show in the remainder of this paper, it is such simplifications in tandem with our problem- rather than algorithm-oriented method of analysis that will allow us to focus on and investigate what we believe to be under-appreciated underpinnings of the computational difficulty of swarm control, in particular those arising from the combinatorial choices in swarm control inputs.

3 RESULTS

In this section, we will assess the algorithmic options for our three swarm control problems defined in Section 2.3 relative to the three types of solvability listed in Section 1. For each type of solvability, we shall first give a more formal definition of that type than was given in Section 1 and then list our solvability and unsolvability results relative to that type for each of our swarm control problems. In order to focus on the **[R4(C1)] meaning of and patterns in** our results in this section, proofs of all results as well as a brief introduction to the techniques by which our unsolvability results are proved are given in the appendix to this paper.

Let us first consider exact polynomial-time solvability. An exact polynomial-time algorithm is an algorithm which always produces the correct output for a given input and whose runtime is **[R1(C12)] asymptotically upper-bounded, i.e., upper-bounded when $|x|$ goes to infinity, by $c|x|^{c'}$, where $|x|$ is the size of the input x and c and c' are constants**. A problem that has a polynomial-time algorithm is said to be **polynomial-time tractable**. Polynomial-time tractability is desirable because runtimes increase slowly as input size increases, and hence allow the solution of larger inputs. There are instances of our problems that are polynomial-time solvable.

Result A.1: SelAlg is polynomial-time tractable when $h = t_{\max} = 1$.

In the case of problem SelAlg parameter h is the maximum number of types of robot controller-algorithms in T at the end of each timestep over all timesteps in construction task execution **and parameter $t_{\max}$ is the last timestep at which controller-algorithm changes are made**. The above result is heartening, as we have quick solvability in the common case when all changes must be made initially at the same time, i.e., $t_{\max} = 1$, such that the robots in the swarm have been changed to have the same controller, i.e., $h = 1$. Unfortunately, this does not hold in general.[6]

Result A.2: SelAlg is not polynomial-time tractable when $h = 2$ and $t_{\max} = 1$ and SelEnvInf and SelLead are not polynomial-time tractable when $h = t_{\max} = 1$.

In the cases of problems SelEnvInf and SelLead, $t_{\max}$ denotes the last timestep at which environment and robot-position changes, respectively, are made. [R2(C5)] It is worth noting

[6] The following four results hold relative to some combination of the conjectures $P \neq NP$ and $P = BPP$, which though unproven are widely believed to be true within computer science [16, 18, 68].

that Results A.1 and A.2 together describe a sharp frontier of polynomial-time tractability (in the sense described in Section 1 and Footnote 1) for SelAlg, SelEnvInf, and SelLead relative to numerical restrictions on the values of h and $t_{\max}$.

It is possible that the computational difficulty of our problems may be inflated in general by inputs that have no solutions, and hence force any algorithm to exhaustively consider all possible candidate solutions. This is addressed by **[R4(C1)] polynomial-time promise solvability**, which is when a problem is exactly solvable in polynomial time on those inputs which are guaranteed to have solutions. However, this is not the case for our problems either.

Result A.3: SelAlg when $h = 2$ and $t_{\max} = 1$ and SelEnvInf and SelLead when $h = t_{\max} = 1$ are not polynomial-time promise solvable.

Let us now consider **[R4(C1)] polynomial-time approximate solvability**. This type of solvability may be acceptable in situations where always getting the correct output for an input is not required. Two popular types of polynomial-time approximation algorithms are those that are frequently correct in that they produce the correct output for a given input either (1) in all but a small number of cases (i.e., the number of errors for input size n is bounded by function $err(n)$) [22] or (2) with a high probability [35]. Unfortunately, these options are not open to us either **[R1(C3)] courtesy of the following two results, which are corollaries of Results A.2 and A.3 and various widely-believed complexity-theoretic assumptions**.

Result A.4: SelAlg when $h = 2$ and $t_{\max} = 1$ and SelEnvInf and SelLead when $h = t_{\max} = 1$ are not solvable by polynomial-time algorithms with polynomial error frequencies (i.e., $err(n)$ is upper bounded by a polynomial of n).

Result A.5: SelAlg when $h = 2$ and $t_{\max} = 1$ and SelEnvInf and SelLead when $h = t_{\max} = 1$ are not polynomial-time solvable by probabilistic algorithms which operate correctly with probability $\geq 2/3$.

[R3(C3)] The above results effectively rule out many desirable forms of efficient solvability for the general unrestricted versions of our swarm control problems. This makes it worthwhile to investigate under what restrictions our problems are efficiently solvable. We will focus here on exact solvability. Let us characterize restrictions on inputs in terms of a set $K = \{k_1, k_2, \ldots, k_{|K|}\}$ of aspects of the input. For instance, some sets of restrictions on the inputs of SelAlg could be $\{|T|\}$, $\{|T|, |E_T|\}$, and $\{h, |f|, r\}$. Each such aspect is also known as a **parameter**. Let $\langle K \rangle$-Π denote a problem Π so restricted relative to an aspect-set K. There are two popular ways in which an algorithm can exactly solve such restricted problems in close-to-polynomial time:

(1) **Fixed-parameter (fp-)tractability** [10]: Such an algorithm runs in time that is non-polynomial purely in terms of the aspects in K, i.e., in time **[R1(C12)] that is asymptotically upper-bounded by** $f(K)|x|^c$ where $f()$ is some function, $|x|$ is the size of input x, and c is a constant. A problem Π with such an algorithm for aspect-set K is said to be **fixed-parameter tractable relative to** K, i.e., $\langle K \rangle$-Π is fixed-parameter tractable. Though they run in non-polynomial time in general, for inputs in which all the aspects in K have very small constant values (and $f(K)$ collapses to a possibly large but nonetheless constant value), such algorithms (particularly if $f()$ is suitably well-behaved, e.g., $(1.2)^{k_1+k_2}$) may be acceptable.

(2) $|x|^k$**-tractability**: Such an algorithm runs in time that is polynomial if the values of all aspects in K have constant value, i.e., in time **[R1(C12)] that is asymptotically upper-bounded by** $|x|^{f(K)}$ for some function $f()$.[7] A problem Π with such an algorithm for aspect-set K is

[7] **[R1(C4)] Note that this type of algorithm defines membership of a problem restricted relative to K in the class XP [10, Page 341].**

Table 1. Parameters for Swarm Control Problems.

Parameter	Description	Applicability
$\|T\|$	# robots in swarm	All
h	# robot-types in swarm	All
$\|Q\|$	Max # states per robot	All
d	Max # transitions per state	All
$\|f\|$	Max transition formula length per robot	All
r	Max robot perceptual radius	All
$\|E\|$	# squares in environment	All
$\|E_T\|$	# distinct environment-square types	All
$\|X\|$	# squares in structure	All
$t_{\max}$	Max change timestep	All
$\|L\|$	# robot algorithms in library	SelAlg
k_a	Max # robot algorithm changes	SelAlg
k_e	Max # environmental changes	SelEnvInf
k_l	Max # robot leader position changes	SelLead

said to be $|x|^k$**-tractable relative to** K, i.e., $\langle K \rangle$-Π is $|x|^k$-tractable. Though they run in non-polynomial time in general, for inputs in which all the aspects in K have very small constant values (and $f(K)$ collapses to a very small constant value), such algorithms (particularly if $f()$ is suitably well-behaved, e.g., $k_1 + k_2$ or $\log_2(k_1 \times k_2)$) may be acceptable.

These types of tractability generalize polynomial-time solvability by allowing either the leading constant c_1 or the exponent c_2 of the input size in the runtime upper-bound of an algorithm to be a function of K rather than a constant. Though fixed-parameter tractability may initially seem preferable in cases where $|x|$ is large, $|x|^k$-tractability may perform better if the exponent-function of $|x|$ is especially well-behaved, e.g., $f(K)$ is either a sublinear function like $\log_2 \log_2(k_1 \times k_2)$ or a small constant.

In our analyses below, we will focus on aspect-sets K drawn from the set of parameters in Table 1. These parameters can be divided into three main groups (the first two of which have previously been defined and analyzed with respect to other swarm robotics problems in [50, 60, 62, 64]):

(1) Parameters characterizing the swarm T and its member robots ($|T|$, h, $|Q|$, d, $|f|$, r,);
(2) Parameters charactering the environment and requested structure ($|E|$, $|E_T|$, $|X|$); and
(3) Parameters characterizing the degree of swarm control ($t_{\max}$, $|L|$, k_a, k_e, k_L).

A proved parameterized tractability or intractability result often implies many others courtesy of the following **[R4(C1)] four lemmas**.

Lemma 3.1. *[58, Lemma 2.1.30] If problem* Π *is fp-tractable relative to aspect-set* K *then* Π *is fp-tractable for any aspect-set* K' *such that* $K \subset K'$.

Lemma 3.2. *[58, Lemma 2.1.31] If problem* Π *is fp-intractable relative to aspect-set* K *then* Π *is fp-intractable for any aspect-set* K' *such that* $K' \subset K$.

Lemma 3.3. *If problem* Π *is* $|x|^k$*-tractable relative to aspect-set* K *then* Π *is* $|x|^k$*-tractable for any aspect-set* K' *such that* $K \subset K'$.

Lemma 3.4. *If problem* Π *is* $|x|^k$*-intractable relative to aspect-set* K *then* Π *is* $|x|^k$*-intractable for any aspect-set* K' *such that* $K' \subset K$.

Let us first consider when each of our problems are and are not fixed-parameter tractable.[8]

- Algorithm Selection:

  Result B.SA.1: $\langle |T|, h, |Q|, |f|, r, |X|, t_{\max}, k_a \rangle$-SelAlg is not fp-tractable.

  Result B.SA.2: $\langle |T|, h, |Q|, |E_T|, |X|, t_{\max}, k_a \rangle$-SelAlg is not fp-tractable.

  Result B.SA.3: $\langle h, |Q|, d, |f|, r, |E_T|, t_{\max}, |L|, k_a \rangle$-SelAlg is not fp-tractable.

  Result B.SA.4: $\langle |T|, h, |Q|, |f|, r, |E|, |X|, t_{\max}, k_a \rangle$-SelAlg is not fp-tractable.

  Result B.SA.5: $\langle |T|, h, |Q|, d, r, |E|, |X|, t_{\max}, k_a \rangle$-SelAlg is not fp-tractable.

  Result B.SA.6: $\langle |T|, h, |f|, |L|, |E_T|, |X|, k_a \rangle$-SelAlg is not fp-tractable.

  Result B.SA.7: $\langle |T|, h, |Q|, d, |L|, |E_T|, |X|, k_a \rangle$-SelAlg is not fp-tractable.

  Result B.SA.8: $\langle |T|, |L|, t_{\max} \rangle$-SelAlg is fp-tractable.

  Result B.SA.9: $\langle |T|, |Q|, r, |E_T|, t_{\max} \rangle$-SelAlg is fp-tractable.

- Environmental Influence Selection:

  Result B.SE.1: $\langle |T|, h, d, |f|, |E_T|, |X|, t_{\max} \rangle$-SelEnvInf is not fp-tractable.

  Result B.SE.2: $\langle |T|, h, |Q|, d, |E_T|, |X|, t_{\max} \rangle$-SelEnvInf is not fp-tractable.

  Result B.SE.3: $\langle |T|, h, |Q|, |f|, r, |E|, |X|, t_{\max}, k_e \rangle$-SelEnvInf is not fp-tractable.

  Result B.SE.4: $\langle |T|, h, |Q|, d, r, |E|, |X|, t_{\max}, k_e \rangle$-SelEnvInf is not fp-tractable.

  Result B.SE.5: $\langle |E|, |E_T|, t_{\max} \rangle$-SelEnvInf is fp-tractable.

- Leader Selection:

  Result B.SL.1: $\langle h, |Q|, d, |E_T|, |X|, t_{\max}, k_l \rangle$-SelLead is not fp-tractable.

  Result B.SL.2: $\langle h, |f|, |E_T|, |X|, t_{\max}, k_l \rangle$-SelLead is not fp-tractable.

  Result B.SL.3: $\langle h, d, |f|, |E_T|, |X|, t_{\max} \rangle$-SelLead is not fp-tractable.

  Result B.SL.4: $\langle |T|, h, |f|, r, |E|, |X|, k_l \rangle$-SelLead is not fp-tractable.

  Result B.SL.5: $\langle |T|, t_{\max} \rangle$-SelLead is fp-tractable.

Next we consider when each of our problems are and are not $|x|^k$-tractable.[9]

[8] Our fp-intractability results hold relative to the conjectures $P \neq NP$ and $FPT \neq W[1]$, which though unproven are both widely believed to be true within computer science [11, 16, 18].

[9] Our $|x|^k$-intractability results hold relative to the conjecture $P \neq NP$, which though unproven is widely believed to be true within computer science [16, 18].

- Algorithm Selection:

  Result C.SA.1: $\langle h, |Q|, d, |f|, r, |E_T|, t_{\max}, |L| \rangle$- SelAlg is not $|x|^k$-tractable.

  Result C.SA.2: $\langle |T|, h, |f|, |X| \rangle$-SelAlg is not $|x|^k$-tractable.

  Result C.SA.3: $\langle |T|, h, |Q|, d, |X| \rangle$-SelAlg is not $|x|^k$-tractable.

  Result C.SA.4: $\langle |T|, h, |f|, |E_T|, |X|, |L| \rangle$-SelAlg is not $|x|^k$-tractable.

  Result C.SA.5: $\langle |T|, h, |Q|, d, |E_T|, |X|, |L| \rangle$-SelAlg is not $|x|^k$-tractable.

  Result C.SA.6: $\langle k_a \rangle$-SelAlg is $|x|^k$-tractable.

- Environmental Influence Selection:

  Result C.SE.1: $\langle |T|, h, d, |f|, |E_T|, |X|, t_{\max} \rangle$-SelEnvInf is not $|x|^k$-tractable.

  Result C.SE.2: $\langle |T|, h, |Q|, d, |E_T|, |X|, t_{\max} \rangle$-SelEnvInf is not $|x|^k$-tractable.

  Result C.SE.3: $\langle k_e \rangle$-SelEnvInf is $|x|^k$-tractable.

- Leader Selection:

  Result C.SL.1: $\langle h, |Q|, d, |E_T|, |X|, t_{\max} \rangle$-SelLead is not $|x|^k$-tractable.

  Result C.SL.2: $\langle h, d, |f|, |E_T|, |X|, t_{\max} \rangle$-SelLead is not $|x|^k$-tractable.

  Result C.SL.3: $\langle |T|, h, |f|, |X| \rangle$-SelLead is not $|x|^k$-tractable.

  Result C.SL.4: $\langle k_l \rangle$-SelLead is $|x|^k$-tractable.

A tighter type of $|x|^k$-intractability result is potentially derivable relative to parameters k_a, k_e, and k_l. In certain cases involving single parameters k, techniques can be applied (see page 1348 of [6] and references) to reduce the $f(k)$ exponents in $|x|^k$-tractable algorithm runtimes to much less than a strictly linear dependence on k, e.g., from $f(k) = k$ to $f'(k) = \log_2 \log_2 k$. A useful characterization of such sublinear behaviour is defined as follows.[10]

Definition 3.5. (Adapted from Definition 3.22 in [14]) Let $f()$ and $g()$ be computable functions. We say that $f(k) \in o(g(k))$ if there is a computable function $h()$ such that for all $\ell \geq 1$ and $k \geq h(\ell)$, we have $f(k) \leq \frac{g(k)}{\ell}$.

It would be very useful if our algorithms above in Results C.SA.6, C.SE.3, and C.SL.4 could be improved in this manner. However, we have the following.[11]

Result C.SA.7: $\langle k_a \rangle$-SelAlg is not $|x|^k$-tractable such that $f(k_a) = o(k_a)$,

[10] This "effective" characterization of sublinear behaviour, though more complex than the usual limit-based formulation of $o()$, i.e., $f(k)$ is $o(g(k))$ if $\lim_{k \to \infty} \frac{g(k)}{f(k)} = 0$, is both technically necessary to handle subtleties introduced when assessing algorithm runtimes relative to different input size measures [14, pages 417–418] and for our purposes equivalent to the limit-based formulation of $o()$.

[11] The following three results hold relative to the conjecture that the Exponential Time Hypothesis is true, which though unproven is widely believed to be true within computer science [25, 26].

Result C.SE.4: $\langle k_e \rangle$-SelEnvInf is not $|x|^k$-tractable such that $f(k_e) = o(k_e)$,
Result C.SL.5: $\langle k_l \rangle$-SelLead is not $|x|^k$-tractable such that $f(k_l) = o(k_l)$.

A final note is in order regarding our intractability results: though we assume bounds of $c_1 = 10$ and $c_2 = 3$ for (c_1, c_2)-completability of construction tasks in Section 2, all of our intractability results hold when $c_1 = 1$ and $c_2 \leq 3$, with most results holding when $c_2 = 1$. Hence, the intractability of our problems does not require exorbitant swarm operation times.

4 DISCUSSION

In this section, we will first summarize our results and consider their completeness and generality (Sections 4.1 and 4.2, respectively). We will then look at the specific implications of these results for both swarm control assistance software tools (Section 4.3) and human control of swarms (Section 4.4). As part of the latter, we will describe how computational complexity analysis (and in particular parameterized complexity analysis) can be used to create a detailed and comprehensive assessment of human cognitive effort for swarm control.

4.1 Completeness of Our Results

Let us first consider the completeness of our results ordered by type of solvability. Our results with respect to exact polynomial-time solvability give us our first surprise. Though SelAlg is polynomial-time tractable when $h = t_{\max} = 1$, i.e., the final swarm is homogeneous and all controller changes required to make it so happen in the first timestep (and hence there are no issues related to change timing) (Result A.1), SelAlg is not polynomial-time tractable when $h = 2$ and $t_{\max} = 1$, and neither are SelEnvInf or SelLead when $h = t_{\max} = 1$ (Result A.2). That this intractability holds at the lowest possible values of h and $t_{\max}$ emphasizes that swarm heterogeneity and timing of swarm inputs, both of which have been assumed to be central to the difficulty of swarm control ([28, page 16] and [28, Section III-F], respectively), are in fact not — intractability remains even if these (and, as noted at the end of Section 2.3, issues related to human-swarm communication and swarm state estimation and prediction by human operators in the face of incomplete observations) are factored out. Moreover, this intractability is not a product of "bad actor" instances of our problems, which have no solutions and hence force needless full searches of the solution space, but holds even for instances where solutions are known to exist (Result A.3). Our results thus expose what **we believe to be an under-appreciated** factor in the intractability of swarm control — namely, the combinatorial choice in swarm control inputs. **This is perhaps not surprising, given that the swarm- and environment-design problems DesCon and EnvDes from [60, 62] (in which combinatorial choice is arguably more obvious) are intractable and special cases of problems SelAlg and SelEnvInf analyzed here.** This combinatorial choice is also powerful enough to rule out both frequently deterministic and frequently probabilistic types of polynomial-time approximate solvability (Results A.4 and A.5, respectively). [12]

The situation is more optimistic with respect to fixed-parameter and $|x|^k$ solvability. Our results show that all three of our problems are fp-intractable, both relative to each of the parameters in Table 1 and in many combinations of these parameters (Results B.SA1–B.SA.7, B.SE.1–B.SE.4, and B.SL.1–B.SL.4); moreover, this fp-intractability holds when many of the parameters have constant values (see Tables 2–5). That being said, we do have several combinations of parameters that yield fp-tractability for each of our problems (Results B.SA.8, B.SA.9, B.SE.5, and B.SL.5). In the case

[12] An examination of the proofs of these results in the appendix shows that they are direct consequences of the polynomial-time intractability of our problems. That such popular types of fast heuristic algorithms can be ruled out so easily is well known within but not outside computational complexity circles [22, 35]. This gives another reason, in addition to others mentioned in the main text, for performing computational complexity analyses.

Table 2. A Detailed Summary of Our Fixed-parameter and $|x|^k$ Complexity Results. a) Results for problem SelAlg. Each column in this table is a result which 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 that parameter in that result's column. Note that within each result-group, intractability results are first and tractability results (shown in bold) are last.

	Fixed-parameter								
]	SA.1	SA.2	SA.3	SA.4	SA.5	SA.6	SA.7	**SA.8**	**SA.9**
$\|T\|$	@	@	–	1	1	1	1	**@**	**@**
h	@	@	2	1	1	1	1	**–**	**–**
$\|Q\|$	3	3	1	@	2	–	2	**–**	**@**
d	–	–	3	–	1	–	2	**–**	**–**
$\|f\|$	16	–	38	3	–	3	–	**–**	**–**
r	1	–	1	@	@	–	–	**–**	**@**
$\|E\|$	–	–	–	@	@	–	–	**–**	**–**
$\|E_T\|$	–	12	6	–	–	4	4	**–**	**@**
$\|X\|$	1	1	–	1	1	1	1	**–**	**–**
$t_{\max}$	1	1	1	@	@	–	–	**@**	**@**
$\|L\|$	–	–	2	–	–	1	1	**@**	**–**
k_a	@	@	@	@	@	@	@	**–**	**–**

Table 3. A Detailed Summary of Our Fixed-parameter and $|x|^k$ Complexity Results (Cont'd). a) Results for problem SelAlg (Cont'd).

	$\|x\|^k$					
	SA.1	SA.2	SA.3	SA.4	SA.5	**SA.6**
$\|T\|$	–	1	1	1	1	**–**
h	2	1	1	1	1	**–**
$\|Q\|$	1	–	2	–	2	**–**
d	3	–	1	–	2	**–**
$\|f\|$	38	3	–	3	–	**–**
r	1	–	–	–	–	**–**
$\|E\|$	–	–	–	–	–	**–**
$\|E_T\|$	6	–	–	4	4	**–**
$\|X\|$	–	1	1	1	1	**–**
$t_{\max}$	1	–	–	–	–	**–**
$\|L\|$	2	–	–	1	1	**–**
k_a	–	–	–	–	–	**@**

of $|x|^k$-tractability, intractability holds relative to a number of combinations of the parameters in Table 1 (Results C.SA.1–C.SA.5, C.SE.1, C.SE.2, and C.SL.1–C.SL.3). Though we do have tractability relative to each of k_a, k_e, and k_l (Results C.SA.6, C.SE.3, and C.SL.4), tractability in the most useful case when the exponent functions in the algorithm runtimes are sublinear seems unlikely (Results C.SA.7, C.SE.4, and C.SL.5).

Table 4. A Detailed Summary of Our Fixed-parameter and $|x|^k$ Complexity Results (Cont'd). b) Results for problem SelEnvInf.

	Fixed-parameter					$\|x\|^k$		
	SE.1	SE.2	SE.3	SE.4	**SE.5**	SE.1	SE.2	**SE.3**
$\|T\|$	1	1	1	1	–	1	1	–
h	1	1	1	1	–	1	1	–
$\|Q\|$	–	1	@	1	–	–	1	–
d	2	2	–	2	–	2	2	–
$\|f\|$	5	–	3	–	–	5	–	–
r	–	–	@	@	–	–	–	–
$\|E\|$	–	–	@	@	**@**	–	–	–
$\|E_T\|$	5	5	–	–	**@**	5	5	–
$\|X\|$	1	1	1	1	–	1	1	–
$t_{\max}$	1	1	1	1	**@**	1	1	–
k_e	–	–	@	@	–	–	–	**@**

Table 5. A Detailed Summary of Our Fixed-parameter and $|x|^k$ Complexity Results (Cont'd). c) Results for problem SelLead.

	Fixed-parameter					$\|x\|^k$			
	SL.1	SL.2	SL.3	SL.4	**SL.5**	SL.1	SL.2	SL.3	**SL.4**
$\|T\|$	–	–	–	1	**@**	–	–	1	–
h	1	1	1	1	–	1	1	1	–
$\|Q\|$	1	–	–	–	–	1	–	–	–
d	3	–	4	–	–	3	4	–	–
$\|f\|$	–	3	3	3	–	–	3	3	–
r	–	–	–	@	–	–	–	–	–
$\|E\|$	–	–	–	@	–	–	–	–	–
$\|E_T\|$	5	5	5	–	–	5	5	–	–
$\|X\|$	1	1	1	1	–	1	1	1	–
$t_{\max}$	1	1	1	–	**@**	1	1	–	–
k_l	@	@	–	–	–	–	–	–	**@**

Though a number of additional fp- and $|x|^k$ tractability and intractability results can be derived from our given results courtesy of Lemmas 3.1–3.4, **[R2(C5)] our results are incomplete in two senses:**

(1) The frontiers of polynomial-time and fp-tractability for SelAlg, SelEnvInf, and SelLead relative to many combinations of parameters in Table 1 are not sharp in the sense described in Section 1 and Footnote 1, in that there are in many cases numerical gaps between the values of parameters for which these problems are known to be tractable and intractable. This is obvious from a perusal of Tables 2–5 — though all combinations of parameters which have an fp-intractability result in which all parameters have value 1 describe sharp (albeit trivial) frontiers of polynomial-time

and fp-tractability, e.g., $\{|T|, h, d\}$ for SelAlg (Result B.SA.5), this is not the case when fp-intractability is only known to hold for combinations of parameters whose values are all greater than 1, e.g., $\{|f|, |E_T|\}$ for SelAlg (Result B.SA.6).

(2) There are a number of combinations of parameters in Table 1 whose fp– and $|x|^k$-complexity have not yet been determined.

However, even these incomplete results allow a much more nuanced view of the computational difficulty of swarm control **[R2(C4)] for distributed construction**. For now, this is via fp-tractability results that are **minimal**, i.e., fp-tractability results $\langle K \rangle$-**X** such that **X** is fp-intractable relative to all proper subsets K' of K. Such minimal fp-tractability results are useful because they indicate problem mechanisms that interact with each other (such that restrictions in one can be traded off against not restricting others to maintain intractability, and tractability only occurs if all are restricted together). At present, we have two minimal fp-tractability results:

(1) $\langle |T|, |L|, t_{\max} \rangle$-SelAlg (Result B.SA.8) (courtesy of Results B.SA.1, B.SA.3, B.SA.6, and Lemma 3.2) and
(2) $\langle |T|, t_{\max} \rangle$-SelLead (Result B.SL.5) (courtesy of Results B.SL.1, B.SL.4, and Lemma 3.2).

Choice in swarm control input timing is encoded in parameter $t_{\max}$ and choice in swarm control inputs is encoded in parameters $|T|$ and $|L|$ (observe that in problem SelLead, there are four choices of movement of a selected robot (north, south, east, and west) and hence $|L|$ is effectively four). Our results show that intractability in SelAlg and SelLead arises not from choices of either timing or swarm control inputs but rather a combination of the two; hence, attempts to get tractability by restricting only one of these aspects (see Section 4.3) are doomed to failure. Hints of other mechanism interactions may also be implicit in certain pairs of intractability results (as visualized in Tables 2–5), where restriction-tradeoffs between pairs of parameter-sets that maintain intractability seem to be occurring, e.g., $\{|T|, h\}$ vs. $\{|Q|, d, |f|, r, |L|\}$ for SelAlg (Results B.SA.2 and B.SA.3), $\{|Q|\}$ vs. $\{r\}$ for SelEnvInf (Results B.SE.1 and B.SE.2), $\{|Q|, d\}$ vs. $\{|f|\}$ for SelLead (Results B.SL1. and B.SL.2). It would be most interesting to see if this interaction can be verified, either via minimal fp-tractability or some other recognizable tractability or intractability result-pattern.

These issues suggest that it would be most useful to **first** complete our analyses by determining the fp-status of our problems relative to all combinations of parameters in Table 1, i.e., performing a **systematic parameterized complexity analysis (SPCA)** [58] for each of our problems. Experience has shown that the effort involved in performing such SPCA can be minimized courtesy of Lemmas 3.1 and 3.2 by deriving fp-tractability and fp-intractability results relative to the smallest and largest possible sets of parameters, respectively. The benefits in turn would be many, including not only resolution of mechanism interaction issues mentioned above (the most notable of which is the fp-minimality of $\langle |E|, |E_T|, t_{\max} \rangle$-SelAlg (Result B.SE.5)) but also determining if other parameter-restricted situations are fp-tractable in addition to those we know of now, as they would yield new and possibly much better algorithms for our problems. **[R2(C5)] Once this SPCA is done, we can then more profitably address the issue of sharpening the frontiers of polynomial-time and fp-tractability for our problems.**

4.2 Generality of Our Results

Let us now consider the generality of our tractability and intractability results. A valid objection is that these results are of limited use as they were derived relative to simplified models. However, as noted previously in [60], it turns out that these results have a broad applicability because our models are special cases of a number of more realistic models (see [60, Section 5] for details), e.g.,

- Our 2D grid-based environments are special cases of 3D grid-based environments.

- 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.
- Our deterministic FSR model is a special case of probabilistic FSR models.
- 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 swarm operation model which does not allow direct communication between robots is a special case of swarm 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 our problems can be created by replacing any combination of simple special-case models with the more general models above. Courtesy of this special-case relationship, 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. Intractability results for Π then also apply to Π' as well as the operation of any automated system solving Π' (see [60, Section 5] for details).[13] Tractability results typically do not propagate from special cases to more general problems, as algorithms often exploit particular details of the inputs and outputs in their associated problems to attain efficiency or even work at all. That being said, some of our tractability results also have a surprisingly broad applicability because the algorithms described in their proofs depend only on the combinatorics of the number of possible swarm control input choices.

Though all of this applicability noted here is for now limited to simple systems in which swarms move in a non-continuous deterministic manner in grid-based environments **[R2(C4)] and perform construction tasks**, there is no reason to expect that our analyses cannot be extended to **[R2(C4)] more realistic swarm models and swarm control in the service of non-construction tasks, e.g., foraging, consensus, navigation** (for further discussion on this point, see Section 5.2 in [60]). **[R2(C4)] The latter is particularly important, as such tasks are typically the focus of swarm control research to date. Our results have already made the following first steps in this:**

- **Consider the navigation task of determining if a swarm can move from a specified initial to a specified final positioning in an environment E. As the proofs of Lemmas 7, 8, 13–16, and 17–19 in the appendix only invoke construction to indicate that the members of a swarm have reached a desired location in E, all intractability results derived relative to those lemmas (Results A.2–A.5, B.SA.3, B.SA.8, B.SE.1–4, B.SL.1–3, C.SA.3, C.SE.1, C.SE.2, C.SE.4, C.SL.1,C.SL.2, and C.SL.5) apply to swarm control for this navigation task.**
- **As all of our tractability results (Results, A.1, B.SA.9, B.SE.5, B.SL.5, C.SA.6, B.SE.3, and C.SL.5) rely only the combinatorics of robot swarm design, environment design, and swarm control input choice, they apply not only to swarm control for this navigation task but to swarm control for *all* possible swarm tasks.**

[13] Additional generalities of our intractability results can be derived by scrutinizing the proofs of these results. For example, swarms constructed in our proofs have either (1) $|T| = 1$ such that synchronous and asynchronous swarm operation are the same or (2) robot controller algorithms and environments which ensure that completable construction tasks eventually finish regardless of the order in which individual robots move, all of our intractability results also hold when swarm operation is asynchronous.

This is not to say that the full pattern of intractability results we have observed relative to the distributed construction task and the navigation task described above necessarily holds for simpler task such as consensus or foraging; indeed, we suspect it does not. However, this can and should be investigated using computational complexity analysis, to establish the frontiers of polynomial-time tractability for swarm control with respect to the types of tasks being performed. It is our hope that the analysis framework we have used and at least some of the proof techniques we have developed will be of use in this endeavor.

In any case, as **was pointed out in Section 4.1**, our simplified model have already proven most useful in allowing us (as was promised at the end of Section 2.3) to explore the sources of computational difficulty in swarm control **[R2(C4)] for distributed construction**. In the following two subsections, we will see how such explorations in turn have implications for both the design and deployment of software assistance tools for swarm control and the investigation of feasible options for the human control of swarms.

4.3 Implications for Swarm Control Assistance Software Tools

Modulo concerns about the simplifications in our models relative to real-world robotics, our tractability results have implications for the design and deployment of software tools to assist human operators in swarm control. In order to maintain user attention, it is known that the time taken by a software system to provide requested advice to humans users should be under 10 seconds [38]. As all of our problems are polynomial-time intractable in general (Result A.2) and do not even allow fast heuristic algorithms that are frequently correct in the most desirable deterministic or probabilistic senses (Results A.4 and A.5, respectively), we need to look at restricted swarm control situations for algorithm runtime efficiency. At present, our fixed-parameter and $|x|^k$ tractability results suggest several such options:

(1) For Algorithm Selection, small swarms with small values of $t_{\max}$ and either small controller-algorithm libraries (Result B.SA.8) or simple robot controllers (Result B.SA.9).
(2) For Environmental Influence Selection, small environments with few square-types and small values of $t_{\max}$ (Result B.SE.5).
(3) For Leader Selection, few selected leaders and small values of $t_{\max}$ (Result B.SL.5).
(4) For all problems, (very) small number of swarm control inputs (Results C.SA.6, C.SE.3, and C.SL.4).

Options 1–3 would work best either for small overall swarms or designated subswarms of larger swarms performing temporally (in the case of Algorithm and Leader Selection) or spatially (in the case of Environmental Influence Selection) limited tasks. Indeed, such algorithms would be very useful in implementing control of large swarms by instead controlling collections of independent subswarms [28, pages 20–22]. Regardless of other factors, if it is known that very few swarm control inputs need to be suggested (as may be the case in temporally or spatially limited tasks), option 4 may be best.

Our analyses can also be used to analyze and mitigate encountered difficulties with existing software tools for swarm control. For example, consider the tool designed in [32, 36] to address the following problem [36, page 2674]:

> *Given a library of swarm behaviors* **B** *and an objective function or performance criterion encoding the task at hand, find the sequence and behaviors and the times at which the behaviors should be switched to accomplish the desired task.*

This tool actually solves the simpler version of this problem in which the set **T** of times at which behaviors can be switched are given as part of the input. Though this tool gives optimal behaviour

sequences relative to the given performance criterion [36, Theorem 2], it was too slow for real-time supervisory control of a real-world swarm [36, page 2681]. This problem was just barely mitigated (i.e., tool latency times were reduced to 2–8 seconds) by extensive parallelization and GPU programming when $|\mathbf{B}|= 7$ [32, page 3808], and even then, human participants in experiments were mostly unwilling to use the sequencing algorithm and found it less usable then generating the sequences manually [32, page 3811]. **[R2(C4)] To us, these difficulties are not surprising, as the proof of Result B.SA.4 can be easily modified to show that the problem solved by the tool in [32, 36] is polynomial-time intractable (albeit for a distributed construction task under our simplified swarm-operation model).** This suggests that a complement to algorithm optimization efforts such as those described in [32] would be to perform a **computational complexity** analysis like those in this paper to establish other possibly more useful algorithm options. Such algorithms for restricted swarm control situations might be a valuable part of proposed research involving incremental construction of behaviour change sequences [32, page 3811]. This analysis could also include looking at speeding up the bounded sub-optimal algorithm sketched on page 2677 of [36] under parameter restrictions using techniques described in [33].

A final note is in order here regarding the admittedly impractical runtimes of the algorithms underlying our presented fixed-parameter **and** $|x|^k$ tractability results. As noted previously in [60, Section 5], this is an artifact of the goal of the analyses in our paper, which is to provide a general overview of the options for restricting problems to obtain practical algorithms; 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 [7, 15, 37], 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 [29]. 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 **and** $|x|^k$ tractability results for swarm control given in this paper.

4.4 Implications for Human Control of Swarms

In the previous section we considered the implications of our complexity results for swarm control assistance software tools and observed that such tools face the intractability inherent in controlling multiple autonomous and interacting robots. Instead of trying to fully automate and offload swarm control to a computer program, might it be more realistic for a human controller to tackle these problems effectively and efficiently in real-time? Our intractability results suggests this is unlikely to be possible.

This is so because arguably humans, like any resource-bounded system, cannot solve intractable problems efficiently [54]. While it is commonly believed that humans may be able to solve such problems approximately or "good enough", such claims turn out to be unfounded: of course humans do use various heuristics or other efficient strategies for problem solving, but if a well-defined computational problem such as those analyzed here is computationally intractable (i.e., *NP*-hard (see the Appendix)), then no such efficient strategies, either deterministically or probabilistically, can yield provably good bounds on the degree of approximation [54, 56, 57] (see also Results A.4 and A.5).

Previous work has already considered the cognitive complexity of human control of autonomous swarms [30]. While these analyses are informative about how the amount of global attention needed to be distributed over the different controlled robots, these analyses are also in a sense limited. This is because those analyses effectively assume that deciding how to control is cognitively for free, in that Lewis' big-Oh scheme (see Section 1.1) only expressed the attentional resources demands for executing chosen control actions, and not the computational complexity of the swarm control input selection problem itself. However, we have shown in this paper that the bulk of the computational complexity of swarm control is hidden in this selection problem.

So is human swarm control hopeless? Most certainly not. This is because, as proposed in [52], humans can, at least in principle and under ideal conditions, efficiently solve fixed-parameter tractable parameterized versions of problems such as those listed in Section 4.3. This does, however, impose strong constraints on swarm design and the design of their control interfaces, as they will need to meet exactly those constraints that render the swarm control problem tractable for humans. Analogous to Lewis' guidelines on how to structure swarms to meet attentional resource limits given on pages 162–163 of [30], we propose to design swarms and the control interfaces to meet the cognitive resource limits on swarm control input selection noted in Section 4.3. For the simple scenarios modeled here this means that our swarm control input selection problems need to be constrained such that the human controller selects

(1) from only a handful of controller algorithms (i.e., library size $|L|$) for a handful of robots (i.e., swarm size $|T|$) over a short period of time (i.e., $t_{\max}$) (when doing algorithm selection);
(2) from only a handful of environment square-types (i.e., environment square-type set $|E_T|$) for a handful of environment squares (i.e., environment size $|E|$) over a short period of time (i.e., $t_{\max}$) (when doing environmental influence selection);
(3) from only a handful of robots (i.e., swarm size $|T|$) over a short period of time (i.e., $t_{\max}$) (when doing leader selection); or
(4) a (very) small number of swarm control inputs (i.e., swarm control input limits k_a, k_e, and k_l when doing algorithm, environmental influence, and leader selection, respectively).

This also demonstrates how computational complexity analysis, and in particular parameterized complexity analysis, can be used to create the novel notion of cognitive complexity for swarm control requested on page 23 of [28] that takes into account not only problem aspects such as system size, task difficulty, and levels of automation but also their respective interactions (along the lines sketched in Section 4.1).

The above poses new and most interesting challenges for swarm control interface design. It would also be interesting to explore the links between our computational complexity results and human performance measures in empirical research, both past and future. We think this latter research will enrich the human factors literature on this topic, because as noted by [28, page 23], this literature has to date focused on performance aspects such as fatigue, attention, and task switching costs, whereas the types of complexity results discussed here address more fundamental competence limitations [17].

5 CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have given the first computational complexity analyses of problems associated with algorithm, environmental influence, and leader selection methods for the control of robot swarms performing distributed construction tasks. These analyses have been done relative to a simple model of distributed construction defined in [62] in which swarms of deterministic finite-state robots operate in a synchronous and non-continuous manner in 2D grid-based environments. We have shown that all three of our problems are polynomial-time intractable in general and that they remain

intractable under a number of plausible restrictions (both individually and in many combinations) on robot controllers, environments, target structures, and sequences of swarm control commands. We have also given the first restrictions relative to which these problems are tractable, as well as discussions of the implications of our results for both the design and deployment of swarm control software assistance tools and the human control of swarms. The last of these also describes how the techniques of computational complexity analysis (and in particular parameterized complexity analysis) can be used to assess the overall cognitive effort involved in human control of swarms.

We believe that the following three whimsically named but seriously intended lessons can be drawn from our analyses:

(1) **Simplification pays**: Rather than being a handicap, simplified models not only allow the isolation and investigation of previously under-appreciated sources of computational difficulty in problems (e.g., combinatorial choice in swarm control inputs (Section 4.1)) but are also a good foundation on which to systematically and rationally build more realistic models.
(2) **Interaction matters**: Observed effects in a system may be the result of multiple interacting mechanisms, and explanations of observed effects that only take into account those mechanisms that are most obvious or manipulated in an experiment may be incomplete. The best interpretation of such observations may come from theoretical analyses like parameterized complexity analysis that explicitly acknowledge and allow investigations of interactions among mechanisms in a system along the lines sketched in Section 4.1.
(3) **Evasion is futile**: Attempts to evade computational difficulties in a system by invoking other systems or processes (e.g., mitigating the difficulties of co-ordinating and controlling swarms by invoking intermittent human interaction with swarms) may be counterproductive — computational difficulty seems be conserved across such system modifications, and if it is not dealt with in the original system or process, it will have to be dealt with in the mitigating one. To paraphrase Robert A. Heinlein's famous acronym TANSTAAFL [21], "There Ain't No Such Thing As A Computationally Free Lunch" (TANSTAACFL[14]).

Given the appearance of the phenomena underlying these lessons in other contexts (e.g., the intractability of evolution when invoked to mitigate the problem of optimally adapting cognitive decision mechanisms [40, 41]), these may in fact be lessons not only for swarm control but computational investigations of real-world systems in general.

There are several promising directions for future research.

- In addition to the future work sketched in Section 4.1 associated with completing the parameterized and $|x|^k$ analyses started in this paper relative to the parameters in Table 1, analyses should also be extended to consider new parameters **[R1(C1)] and other forms of efficient solvability. Of particular interest here are parameters that further restrict the structures of swarms, environments and swarm control command sequences such that our intractability proofs are invalidated and recently-proposed types of probabilistic fixed-parameter tractability [9, 34]**.
- **[R2(C4)] In addition to the future work sketched in Section 4.2 on swarm control for non-construction tasks**, more realistic problems should also be formalized and investigated. Aside from problems based on more realistic notions of robot and swarm operation (continuous asynchronous motion in a continuous 3D environment), it would be of interest to look at problems involving other types of swarm control (e.g., environmental influence by the placement of beacons and/or pheromone sources, dividing a large swarm into independent subswarms and co-ordinating the control of these subswarms) and more specific swarm

[14] Suggested pronunciation: tans-TACK-full.

control activities (e.g., prediction of swarm states when (c_1, c_2)-completability of tasks is not guaranteed).

The most important direction for future work, though, is the development of **new and improved algorithms** for real-time control of real-world swarms. **[R4(C5)] Such algorithms will be very useful not only in Human-Robot Interaction [28] but also Collaborative Human-Robot Construction [20]. The development of these algorithms** will involve both classical algorithm engineering for swarm control assistance software tools along the lines sketched in Section 4.3 as well as detailed assessments of the overall cognitive effort of human control of swarms. The latter may best be realized via interleaved rounds of theoretical and experimental analysis in a research framework analogous to the natural language complexity game [42] and the tractable theory revision cycle for cognitive science [53–55]. It is our hope that the techniques and analyses given in this paper will be useful foundations for all of these efforts.

ACKNOWLEDGMENTS

The authors would like to thank the four anonymous reviewers, whose comments helped to improve both the technical content and the presentation of the paper. TW was supported by National Science and Engineering Research Council (NSERC) grant 228105-2015.

REFERENCES

[1] 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.

[2] Michael Allwright, Navneet Bhalla, and Marco Dorigo. 2017. Structure and markings as stimuli for autonomous construction. In *Proceedings of the 2017 18th International Conference on Advanced Robotics*. IEEE, 296–302.

[3] Hadi Ardiny, Stefan Witwicki, and Francesco Mondada. 2015. Are Autonomous Mobile Robots Able to Take Over Construction? A Review. *International Journal of Robotics: Theory and Applications* 4, 3 (2015), 10–21.

[4] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. 1999. *Swarm intelligence: From natural to artificial systems*. Oxford University Press.

[5] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. 2013. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence* 7, 1 (2013), 1–41.

[6] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. 2006. Strong computational lower bounds via parameterized complexity. *J. Comput. System Sci.* 72 (2006), 1346–1367.

[7] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer.

[8] Erik Demaine, Mohammad Hajiaghayi, and Dániel Marx. 2014. Minimizing movement: Fixed-parameter tractability. *ACM Transactions on Algorithms* 11, 2 (2014), 1–29.

[9] Nils Donselaar. 2019. Probabilistic parameterized polynomial time. In *International Conference on Current Trends in Theory and Practice of Informatics*. Springer, 179–191.

[10] Rodney G. Downey and Michael R. Fellows. 1999. *Parameterized Complexity*. Springer, Berlin.

[11] Rodney G. Downey and Michael R. Fellows. 2013. *Fundamentals of Parameterized Complexity*. Springer, Berlin.

[12] Philip E. Dunne, Michael Laurence, and Michael Wooldridge. 2003. Complexity results for Agent Design. *Annals of Mathematics, Computing & Teleinformatics* 1, 1 (2003), 19–36.

[13] 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.

[14] Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Springer, Berlin.

[15] Fedor V. Fomin, Daniel Lokshantov, Saket Saurabh, and Meirav Zehavi. 2019. *Kernalization: Theory of Parameterized Prepreocessing*. Cambridge University Press, Cambridge, UK.

[16] Lance Fortnow. 2009. The Status of the P Versus NP Problem. *Commun. ACM* 52, 9 (2009), 78–86.

[17] Marcello Frixione. 2001. Tractable competence. *Minds and Machines* 11, 3 (2001), 379–397.

[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 Workshop on Foundations and Applications of Self* Systems*. IEEE, 162–167.

[20] Isla Xi Han, Forrest Meggers, and Stefana Parascho. 2021. Bridging the collectives: A review of collective human–robot construction. *International Journal of Architectural Computing* 19, 4 (2021), 1–20.
[21] Robert A. Heinlein. 1966. *The Moon is a Harsh Mistress.* G. P. Putnam's Sons, New York.
[22] Lane A. Hemaspaandra and Ryan Williams. 2012. Complexity Theory Column 76: An atypical survey of typical-case heuristic algorithms. *ACM SIGACT News* 43, 4 (2012), 70–89.
[23] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2001. *Introduction to Automata Theory, Languages, and Computation* (2nd ed.). Addison-Wesley.
[24] 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". *The International Journal of Robotics Research* 3, 4 (1984), 76–88.
[25] Russell Impagliazzo and Ramamohan Paturi. 2001. On the complexity of k-SAT. *Journal of Computer and System Science* 62, 2 (2001), 367–375.
[26] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which problems have strongly exponential complexity? *Journal of Computer and System Science* 63, 4 (2001), 512–530.
[27] Andreas Kolling, Steven Nunnally, and Michael Lewis. 2012. Towards human control of robot swarms. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*. 89–96.
[28] Andreas Kolling, Phillip Walker, Nilanjan Chakraborty, Katia Sycara, and Michael Lewis. 2016. Human interaction with robot swarms: A survey. *IEEE Transactions on Human-Machine Systems* 46, 1 (2016), 9–26.
[29] 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.
[30] Michael Lewis. 2013. Human interaction with multiple remote robots. *Reviews of Human Factors and Ergonomics* 9, 1 (2013), 131–174.
[31] Michael Lewis, Jijun Wang, and Paul Scerri. 2006. Teamwork coordination for realistically complex multi robot systems. In *NATO Symposium on Human Factors of Uninhabited Military Vehicles as Force Multipliers*. 1–12.
[32] Huao Li, Jaeho Bang, Sasanka Nagavalli, Changjoo Nam, Michael Lewis, and Katia Sycara. 2018. Human Interaction Through an Optimal Sequencer to Control Robotic Swarms. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 3807–3812.
[33] Dániel Marx. 2008. Parameterized complexity and approximation algorithms. *Comput. J.* 51, 1 (2008), 60–78.
[34] Juan Andrés Montoya and Moritz Müller. 2013. Parameterized random complexity. *Theory of Computing Systems* 52, 2 (2013), 221–270.
[35] Rajeev Motwani and Prabhakar Raghavan. 2010. *Randomized Algorithms.* Chapman & Hall/CRC.
[36] Sasanka Nagavalli, Nilanjan Chakraborty, and Katia Sycara. 2017. Automated sequencing of swarm behaviors for supervisory control of robotic swarms. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2674–2681.
[37] Rolf Niedermeier. 2006. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press.
[38] Jakob Nielsen. 1994. *Usability Engineering.* Morgan Kaufmann.
[39] Kirstin H. Petersen, Nils Napp, Robert Stuart-Smith, Daniela Rus, and Mirko Kovac. 2019. A review of collective robotic construction. *Science Robotics* 4, 28 (2019), 1–10.
[40] Patricia Rich, Mark Blokpoel, Ronald de Haan, Maria Otworowska, Marieke Sweers, Todd Wareham, and Iris van Rooij. 2021. Naturalism, tractability and the adaptive toolbox. *Synthese* 198, 6 (2021), 5749–5784.
[41] Patricia Rich, Mark Blokpoel, Ronald de Haan, and Iris van Rooij. 2020. How intractability spans the cognitive and evolutionary levels of explanation. *Topics in Cognitive Science* 12, 4 (2020), 1382–1402.
[42] Eric S. Ristad. 1993. *The Language Complexity Game.* MIT Press.
[43] Kamel S. Saidi, Thomas Bock, and Christos Georgoulas. 2016. Robotics in construction. In *Handbook of Robotics*. Springer, 1493–1520.
[44] Thomas B. Sheridan and William L. Verplank. 1978. *Human and computer control of undersea teleoperators.* Technical Report. Massachusetts Institute of Technology Man-Machine Systems Lab.
[45] Touraj Soleymani, Vito Trianni, Michael Bonani, Francesco Mondada, and Marco Dorigo. 2015. Bio-inspired construction with mobile robots and compliant pockets. *Robotics and Autonomous Systems* 74 (2015), 340–350.
[46] Ian A. Stewart. 2003. The complexity of achievement and maintenance problems in agent-based systems. *Artificial Intelligence* 2, 146 (2003), 175–191.
[47] 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. *Autonomous Robots* 20, 2 (2006), 113–123.
[48] Guy Theraulaz and Eric Bonabeau. 1995. Coordination in distributed building. *Science* 269, 5224 (1995), 686.

[49] Skylar Tibbits (Ed.). 2017. Special issue: Autonomous assembly: designing for a new era of collective construction. *Architectural Design* 87, 4 (2017), 1–136.
[50] 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.
[51] Vaibhav V Unhelkar, Przemyslaw A Lasota, Quirin Tyroller, Rares-Darius Buhai, Laurie Marceau, Barbara Deml, and Julie A Shah. 2018. Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time. *IEEE Robotics and Automation Letters* 3, 3 (2018), 2394–2401.
[52] Iris van Rooij. 2008. The Tractable Cognition Thesis. *Cognitive Science* 32, 6 (2008), 939–984.
[53] Iris van Rooij and Giosuè Baggio. 2021. Theory before the test: How to build high-versimilitude explanatory theories in psychological science. *Perspectives on Psychological Science* 16, 4 (2021), 682–697.
[54] Iris van Rooij, Mark Blokpoel, Johan Kwisthout, and Todd Wareham. 2019. *Cognition and Intractability: A Guide to Classical and Parameterized Complexity Analysis.* Cambridge University Press, Cambridge, UK.
[55] Iris van Rooij and Todd Wareham. 2008. Parameterized Complexity in Cognitive Modeling: Foundations, Applications, and Opportunities. *Computer Journal* 51, 3 (2008), 385–404.
[56] Iris van Rooij and Todd Wareham. 2012. Intractability and Approximation of Optimization Theories of Cognition. *Journal of Mathematical Psychology* 56, 4 (2012), 232–247.
[57] Iris van Rooij, Cory D. Wright, and Todd Wareham. 2012. Intractability and the Use of Heuristics in Psychological Explanation. *Synthese* 187, 2 (2012), 471–487.
[58] Todd Wareham. 1999. *Systematic Parameterized Complexity Analysis in Computational Phonology.* Ph.D. Dissertation. University of Victoria, Canada.
[59] 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.
[60] Todd Wareham. 2019. Designing Robot Teams for Distributed Construction, Repair, and Maintenance. *ACM Transactions on Autonomous and Adaptive Systems* 14(1) (2019), 2:1–2:29.
[61] 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.
[62] Todd Wareham and Andrew Vardy. 2018. Putting It Together: The Computational Complexity of Designing Robot Controllers and Environments for Distributed Construction. *Swarm Intelligence* 12, 2 (2018), 111–128.
[63] Todd Wareham and Andrew Vardy. 2018. Viable Algorithmic Options for Designing Reactive Robot Swarms. *ACM Transactions on Autonomous Adaptive Systems* 13, 1 (2018), 5:1–5:23.
[64] Todd Wareham and Andrew Vardy. 2021. The Computational Complexity of Designing Scalar-Field Sensing Robot Teams and Environments for Distributed Construction. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C).* IEEE Press, Los Alamitos, CA, 232–237.
[65] Eric W. Weisstein. 2020. von Neumann Neighborhood. (2020). https://mathworld.wolfram.com/vonNeumannNeighborhood.html From *MathWorld* – A Wolfram Web Resource.
[66] Justin Werfel and Radhika Nagpal. 2006. Extended stigmergy in collective construction. *IEEE Intelligent Systems* 21, 2 (2006), 20–28.
[67] J. Werfel, Kirsten Petersen, and R. Nagpal. 2014. Designing collective behavior in a termite-inspired robot construction team. *Science* 343 (2014), 754–758.
[68] Avi Wigderson. 2007. P, NP and Mathematics — A computational complexity perspective. In *Proceedings of ICM 2006: Volume I.* EMS Publishing House, Zurich, 665–712.
[69] Michael Wooldridge and Philip E. Dunne. 2002. The computational complexity of agent verification. In *Intelligent Agents VIII.* Springer, 115–127.

A PROOFS OF RESULTS

A.1 Preliminaries

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'

[11] allows the instance transformation algorithm to run in fp-time relative to K and requires that (1) for each $k' \in K'$ that 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.[15] 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 following problems:

3-Satisfiability (3SAT) [18, Problem LO2]
Input: A set U of variables and a set C of disjunctive clauses over U such that each clause $c \in C$ has $|c| = 3$.
Question: Is there a satisfying truth assignment for C?

Weighted 3-satisfiability (W3SAT) [10]
Input: A set U of variables, a set C of disjunctive clauses over U such that each clause $c \in C$ has $|c| = 3$, and a positive integer k.
Question: Is there a satisfying truth assignment for C of weight k, i,e, a satisfying truth assignment in which exactly k variables are set to $True$?

Clique [18, Problem GT19]
Input: An undirected graph $G = (V, E)$ and a positive integer k.
Question: Does G contain a clique of size k, i.e., is there a subset $V' \subseteq V$, $|V'| = k$, such that for all $u, v \in V'$, $(u, v) \in E$?

Dominating set [18, Problem GT2]
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 $(v, v') \in E$?

Let Dominating set^{PD3} be the version of Dominating set in which G is planar and each vertex has degree at most 3. The problems 3SAT, Clique, Dominating Set, and Dominating set^{PD3} are NP-hard in general (see problem references in [18] above); moreover problem Clique is $W[1]$-hard and problems W3SAT and Dominating set are $W[1]$-hard and $W[2]$-hard, respectively, relative to aspect-set $\{k\}$ [10]. For each vertex $v \in V$, let the complete neighbourhood $N_C(v)$ of v 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 Clique and Dominating set an arbitrary ordering on the vertices of V such that $V = \{v_1, v_2, \ldots, v_{|V|}\}$; we also assume analogous orderings for the variables and clauses of each instance of 3SAT and W3SAT such that $U = \{u_1, u_2, \ldots, u_{|U|}\}$ and $C = \{c_1, c_2, \ldots, c_{|C|}\}$.

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 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

[15] Note that this definition given here is actually Definition 6.1 in [54], which modifies that in [10] to accommodate parameterized problems with multi-parameter sets.

following four easily-proven lemmas will be useful below in transferring results from decision problems to their associated non-decision and parameterized problems; these lemmas follow from the observation that any algorithm for non-decision problem **X** can be used to solve $\mathbf{X}_D$ and the definitions of fp- and $|x|^k$-tractability.

Lemma A.1. *If $\mathbf{X}_D$ is not solvable in polynomial time relative to conjecture* **C** *then* **X** *is not solvable in polynomial time relative to conjecture* **C**.

Lemma A.2. *Given a parameter-set K for problem* **X**, *if $\langle K \rangle$-$\mathbf{X}_D$ is not fixed-parameter tractable relative to conjecture* **C** *then $\langle K \rangle$-***X** *is not fixed-parameter tractable relative to conjecture* **C**.

Lemma A.3. *Given a parameter-set K for problem* **X**, *if $\langle K \rangle$-$\mathbf{X}_D$ is not $|x|^k$-tractable relative to conjecture* **C** *then $\langle K \rangle$-***X** *is not $|x|^k$-relative to conjecture* **C**.

Lemma A.4. *Given a parameter-set K for problem* **X**, *if $\mathbf{X}_D$ is NP-hard when the value of every parameter $k \in K$ is fixed to a constant value, then $\langle K \rangle$-**X** $\notin$ FPT unless $P = NP$.*

A.2 Proofs

[R2(C6)] This subsection contains the proofs of all results cited in the main text. As noted in Section 1.2 in the main text, proofs of certain Lemmas are either straightforward adaptations of (Lemmas A.5, A.6, and A.13–16) or build on (Lemmas A.9 and A.20) proofs given previously in [60, 62]. In both cases, to avoid repeating the full proof and committing self-plagiarism, we first describe in general how the previous proof operated and then give the changes required for this paper (which in the cases of Lemmas A.9 and A.20 are necessarily more detailed). All other proofs are given here in full detail.

We first address the issue of absolute versus operational FSR determinism discussed in Footnote 4 in the main text. Consider the following problem:

FSR Nondeterminism
Input: An FSR X with state-set Q and sensory radius r, a 2D grid G, and environment square-type set E_T.
Question: Is the operation of X not deterministic in some possible environment based on G and E, in the sense that there is an environment E based on G and E_T and a state $q \in Q$ such that X in state q can be positioned in E and at least two outgoing transitions from q are enabled that do not perform the same environment modifications and progress to the same next state?

Result X.1: If FSR Nondeterminism is polynomial-time tractable then $P = NP$.

Proof. Consider the following polynomial-time reduction from 3SAT to FSR Nondeterminism: Given an instance $\langle C, U \rangle$ of 3SAT, construct an instance $\langle X, G, E_T \rangle$ of FSR Nondeterminism such that G is a $(|U| + 1) \times 1$ 2D grid, $E_T = \{e_T, e_F\}$ and X has two states, q_0 and q_1, and sensory-radius $r = |U|$. An environment E based on G and E_T can encode the variables in U such u_i, $1 \leq i \leq |U|$, corresponds to the square $E_{i+1,1}$. Given such an environment, let X be positioned at $E_{1,1}$ and $E(e)$ be the offset from X of the square corresponding to variable $u \in U$. Given this, let X have the following two transitions:

(1) $\langle q_0, f_1, *, stay, q_1 \rangle$, where f_1 is the propositional formula consisting of the disjunction of all clauses in C such that u ($\neg u$) is replaced by predicate $enval(e_T, E(u))$ ($enval(e_F, E(u))$).
(2) $\langle q_0, f_2, *, stay, q_0 \rangle$, where f_2 is the propositional formula consisting of the disjunction of all predicates in the set $\{enval(e_T, E(u)) \mid u \in U\} \cup \{enval(e_F, E(u)) \mid u \in U\}$.

This instance of FSR Nondeterminism can be constructed in time polynomial in the given instance of 3SAT. To prove the correctness of this reduction, note that (1) the first transition enables if and

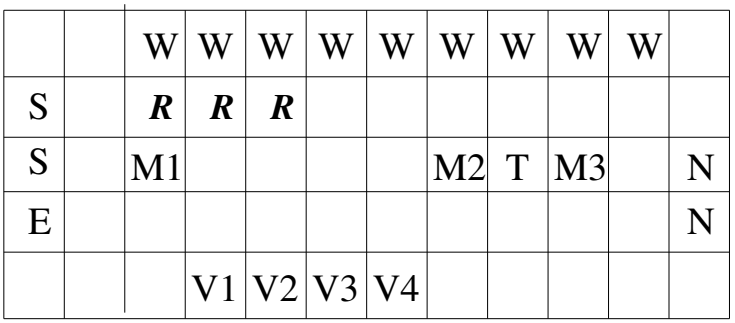

Fig. 1. An example environment and initial placement region E_I of a FSR swarm T constructed in the proof of Lemma A.5 when in the given instance of Dominating set, $|V| = 4$ and $k = 2$ [60, Figure 4] Robot movement (N, S, E, W) and activity (V1, V2, V3, V4, M1, M2, M3, T) square-types are shown in 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.

only if there is a truth assignment for the variables in U encoded in E that satisfies the disjunction of all clauses in C and (2) the second transition enables for *every possible* truth assignment for the variables of U that can be encoded in E. Hence, the two transitions in X can be simultaneously enabled in an environment E and have different effects (i.e., transitions (1) and (2) change the state to q_0 and q_1, respectively) if and only if there is a truth assignment for the variables in U encoded in E that satisfies the disjunction of all clauses in C, and the answer to the given instance of 3SAT is "Yes" if and only if the answer for the constructed instance of FSR Nondeterminism is "Yes". This completes the proof. □

We next consider the swarm control problems defined in Section 2.3.

Lemma A.5. Dominating set *polynomial-time reduces to* $SelAlg_D$ *such that in the constructed instance of* $SelAlg_D$, $|Q| = 3$, $r = |X| = t_{\max} = c_1 = c_2 = 1$, $|f| = 16$, *and* $|T|$, h, *and* k_a *are functions of* k *in the given instance of* Dominating set.

Proof. Adapted from the proof of Lemma A.2 in [60]. In that proof, for a given instance of Dominating set, we construct an instance of SelAlg with an $(|V| + 8) \times 5$ environment like that in Figure 1 such that a swarm of $k + 1$ robots will be able to co-operatively construct a single-square structure at the square with type e_T if and only if there is a dominating set of size at most k in the graph in the given instance of Dominating set. The robots are initially positioned in the north-west 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 swarm consists of at most k "vertex neighbourhood" 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 e_T. Regardless of how these robots on this swarm are initially positioned, the vertex neighbourhood robots will, if they can, have filled in the central scaffolding after the entire swarm has been around the track at most two times, leaving the checker robot to verify their work on the third time around.

In this proof, we initially give all $k+1$ robots a "dud" controller that does nothing, i.e., a controller with the single state q_0 and the single transition $\langle q_0, *, *, stay, q_0 \rangle$. Let L be the union of the vertex neighbourhood, checker, and dud robot controllers, $t_{\max} = 1$, and $k_a = k + 1$. This construction

can be done in time polynomial in the size of the given instance of Dominating set. If there is a dominating set of size k in G in the given instance of Dominating set, C_A can change the controllers of k of the dud robots to the vertex neighbourhood controllers corresponding to the vertices in the dominating set and the controller of the $(k+1)$st robot to the checker controller; as in original lemma, placement of the vertex neighbourhood and checker robots is immaterial. Conversely, if there is a set C_A of controller changes that allows the resulting swarm to create X, the $k+1$ dud robots must have had their controllers changes to a checker controller and k vertex neighbourhood controllers that correspond to a dominating set of size k in G in the given instance of Dominating set. This establishes the correctness of the reduction and completes the proof. □

Lemma A.6. Dominating set *polynomial-time reduces to* $SelAlg_D$ *such that in the constructed instance of* $SelAlg_D$, $|Q| = 3$, $|E_T| = 12$, $|X| = t_{\max} = c_1 = c_2 = 1$, *and* $|T|$, h, *and* k_a *are functions of* k *in the given instance of* Dominating set.

Proof. Adapted from the proof of Lemma A.3 in [60]. In that proof, each square of type e_{vi} in the proof of Lemma A.2 in [60] is replaced by a north-south-oriented sequence of height $\lceil \log_2 |V| \rceil$ composed of squares of type e_0 and e_1 corresponding to a binary encoding of the integer i; in order to correctly recognize these e_{vi}-sequences, all vertex-neighbourhood robots have sensory radius $r = \lceil \log_2 |V| \rceil$ and have their transition-activation formulas modified accordingly. In our proof, we modify L and the initial $k+1$ robots along the lines in the proof of Lemma A.5 above. □

Lemma A.7. *3SAT polynomial-time reduces to* $SelAlg_D$ *such that in the constructed instance of* $SelAlg_D$, $h = 2$, $|Q| = 1$, $d = 3$, $|f| = 38$, $r = c_1 = c_2 = 1$, $|E_T| = 6$, $t_{\max} = 1$, *and* $|L| = 2$.

Proof. Given an instance $\langle C, U, k \rangle$ of 3SAT, construct an instance $\langle E, E_T, T, L, p_I, X, p_X, t_{\max}, k_a \rangle$ of SelAlg_D as follows: Let $E_T = \{e_0, e_T, e_F, e_U, e_B, e_X\}$ and E be based on a $(|U|+2) \times (|C|+1)$ grid in which the southmost $|C|$ squares of the westmost and eastmost columns are of type e_U, the middle $|U|$ squares of the northmost row are of type e_B, each row i, $1 \leq i \leq |C|$, has type e_F (e_T) in square $E_{k+1,i}$ if variable j in clause c_i in C is $\neg u_k$ (u_k) for $1 \leq j \leq 3$, and type e_0 otherwise. Let L consist of two variable-robots, r_F and r_T, where r_F and r_T each have a single state q_0, sensory radius $r = 1$, and the following transitions:

(1) $\langle q_0, (enval(e_U, (-1,0)) \text{ and } enval(e_{robot}, (1,1))) \text{ or}$
$(enval(e_{robot}, (-1,1)) \text{ and } enval(e_U, (1,0))) \text{ or}$
$(enval(e_{robot}, (-1,1)) \text{ and } enval(e_{robot}, (1,0))) \text{ or}$
$(enval(e_{robot}, (-1,0)) \text{ and } enval(e_{robot}, (1,1))) \text{ or}$
$(enval(e_{robot}, (-1,1)) \text{ and } enval(e_{robot}, (1,1))) \text{ or}$
$(enval(e_{robot}, (-1,0)) \text{ and } enval(e_{robot}, (1,0)) \text{ and } enval(e_F, (0,0))),$
$*, goNorth, q_0\rangle$ for r_F and
$\langle q_0, (enval(e_U, (-1,0)) \text{ and } enval(e_{robot}, (1,1))) \text{ or}$
$(enval(e_{robot}, (-1,1)) \text{ and } enval(e_U, (1,0))) \text{ or}$
$(enval(e_{robot}, (-1,1)) \text{ and } enval(e_{robot}, (1,0))) \text{ or}$
$(enval(e_{robot}, (-1,0)) \text{ and } enval(e_{robot}, (1,1))) \text{ or}$
$(enval(e_{robot}, (-1,1)) \text{ and } enval(e_{robot}, (1,1))) \text{ or}$
$(enval(e_{robot}, (-1,0)) \text{ and } enval(e_{robot}, (1,0)) \text{ and } enval(e_T, (0,0))),$
$*, goNorth, q_0\rangle$ for r_T.

(2) $\langle q_0, enval(e_B, (0,0)), enmod(e_X, (0,0)), stay, q_0\rangle$

(3) $\langle q_0, enval(e_X, (0,0)), *, stay, q_0\rangle$

Transition (1) allows a variable-robot x to progress northwards if the variable-robot on either side of x is already one square to the north or the variable-robots on both sides of x are in the same row as x and x is currently on a square of type e_F (e_T) if $x = r_F$ ($x = r_T$). Note that this allows a variable-robot x to progress to the next clause-row in E only if either x or some other variable-robot is set to satisfy the clause in C corresponding to the current row in E in which x is standing. Transition (2) allows a variable-robot x to construct a structure square when x reaches the northmost row in E, and clause (3) ensures that x does nothing thereafter. Let T consist of $|U|$ r_F variable robots initially positioned in the middle $|U|$ squares of the southmost row of E, X consist of a line of $|U|$ squares of type e_X, and p_X specify the middle $|U|$ squares in the northmost row of E. Finally, set $t_{\max} = 1$ and $k_a = |U|$. Note that this instance of SelAlg_D can be constructed in time polynomial in the given instance of 3SAT.

We now need to show the correctness of the reduction above:

- Suppose there is a satisfying truth assignment for c to U in the given instance of 3SAT. Let C_A be the set of controller-changes that converts the controllers of each of the variable-robots in T to r_T (r_F) if the variable in U corresponding to that robot has value $True$ ($False$) in the satisfying truth assignment. By the structure of E, T modified by C_A will progress from the southmost to the northmost row and successfully create requested structure X and p_X. As each robot in T needs to go northwards $|U|$ squares to reach the northmost row and then construct a square of type e_X in its corresponding column in the northmost row, the number of timesteps required for t modified by C_A to construct X at p_X is $\leq c_1|E|^{c_2} = |E|$, which means that the construction task is (c_1, c_2)-completable wrt T and C_A when $c_1 = c_2 = 1$.
- Conversely, suppose there is a set of controller-changes C_A such that for T started at p_I and modified by C_A, the task of constructing X at p_X is (c_1, c_2)-completable. As all variable-robots in T must reach the northmost row in E to construct X, at least one variable-robot in each clause-row in E must have been set to satisfy the corresponding clause in C and hence allow all of the variable-robots in T to progress past that clause-row. This means that the variable-robots in T as modified by C_A correspond to a satisfying truth assignment to U for C in the given instance of 3SAT.

This establishes the correctness of the reduction and completes the proof. □

Lemma A.8. *$\langle k \rangle$-W3SAT parameterized reduces to $\langle h, |Q|, d, f, r, |E_T|, |L|, t_{\max}, k_a \rangle$-$\text{SelAlg}_D$ such that in the constructed instance of SelAlg_D, $h = 2$, $|Q| = 1$, $d = 3$, $|f| = 38$, $r = 1$, $|E_T| = 6$, $t_{\max} = 1$, and $|L| = 2$, and k_a is a function of k in the given instance of W3SAT.*

Proof. Follows (modulo slight changes to the proof of correctness) from the reduction described in Lemma A.7 when $k_a = k$ rather than $k_a = |U|$. □

Lemma A.9. Clique *polynomial-time reduces to SelAlg_D such that in the constructed instance of SelAlg_D, $|T| = h = |X| = c_1 = 1$, $c_2 = 2$, $|f| = 3$, and Q, r, $|E|$, $t_{\max}$, and k_a are functions of k in the given instance of* Clique.

Proof. Inspired by the proof of Lemma 8 in the Supplementary Materials for [62]. In that proof, for a given instance of Clique, we construct an environment E specified in a $2 \times (|k| + 1)$ grid such that $E_{2,1}$ has type e_{F1}, the northmost k squares in the westmost column either have types from the set $\{e_1, e_2, \ldots, e_{|V|}\}$, and all other squares have type e_0. A given robot *rob* initially placed at $E_{1,1}$ is structured such that *rob* can only progress to $p_X = E_{2,1}$ to create the single-square structure X if

the northmost k squares in the westmost column of E specify a clique of size k in G in the given instance of Clique.

In this proof, we have a robot-library consisting of three types of robots:

(1) A dud robot with a single state q_0 and the single transition $\langle q_0, *, *, stay, q_0\rangle$;
(2) $|V|$ vertex robots, each with $r = 0$ and $Q = \{q_0, q_1\}$, such that vertex robot i, $1 \leq i \leq |V|$, has the pair of transitions $\langle q_0, enval(e_0, (0, 0)), enmod(e_{v_i}, (0, 0)), goSouth, q_1\rangle$ and $\langle q_1, *, *, stay, q_1\rangle$; and
(3) A checker robot based on rob with a new initial state q_0 and the additional transition $\langle q_0, enmod(e_{F0}, (0, 0)), e_0, stay, q_0'\rangle$, where q_0' is the initial state of rob.

Let E be such that $E_{1,1} = e_{F0}$ and $E_{2,1} = e_{F1}$ and all other squares have type e_0, T consist of a single dud robot initially positioned at $E_{k+1,1}$, and $t_{max} = k_a = k + 1$. This construction can be done in time polynomial in the size of the given instance of Clique. If there is a clique of size k in G in the given instance of Clique, C_A can change the single initial dud robot in T first to the vertex robots corresponding to the vertices in that clique and then the checker robot. This will ensure that the encoding of that k-clique is correctly written in the northmost k squares of the westmost column of E and then verified to allow the construction of X at $E_{2,1}$. Conversely, if there is a set C_A of robot-type changes to T that allows the checker robot to create X at p_X, the dud robot must first have been changed to k vertex robots (to both create an encoding of a candidate k-clique in G in the given instance of Clique in the northmost k squares in the westmost column of E and move the robot to $E_{1,1}$) and then changed to the checker robot (to verify that the encoding is in fact a k-clique in G and then construct X at $E_{2,1}$). This establishes the correctness of the reduction and completes the proof. □

Lemma A.10. Dominating set *polynomial-time reduces to* $SelAlg_D$ *such that in the constructed instance of* $SelAlg_D$, $|T| = h = d = |X| = c_1 = c_2 = 1$, $|Q| = 2$, *and* r, $|E|$, $t_{\max}$, *and* k_a *are functions of* k *in the given instance of* Dominating set.

Proof. Consider the following reduction from Dominating set to SelAlg_D. Given an instance $\langle G = (V, E), k\rangle$ of Dominating set, construct an instance $\langle E, E_T, T, L, p_I, X, p_X, t_{\max}, k_a\rangle$ of SelAlg_D as follows: Let $E_T, T, p_I, X, t_{\max}, k_a$, and the dud and vertex robots in L be as specified in the reduction in the proof of Lemma A.9. We will modify E to be a $1 \times (k + 1)$ grid in which $E_{1,1}$ has type e_{F0} and all other squares have type e_0, $p_X = E_{1,1}$, and the checker robot in robot L to have a single state q_0 and the single transition $\langle q_0, enval(e_{F0}, (0, 0))$ and $(\mathbf{F})$, $enmod(e_X, (0, 0)), stay, q_0\rangle$, where $\mathbf{F}$ is the conjunction of $|V|$ parenthesis-enclosed formulas, the ith of which for $1 \leq i \leq |V|$ is the disjunction of the predicates in the set $\{enval(e_{vj}, (0, l)) \mid 1 \leq l \leq k$ and $\mathrm{v_j} \in \mathrm{N_C(v_i)}\}$. This construction can be done in time polynomial in the size of the given instance of Dominating set.

Let us now consider the correctness of this reduction. If there is a dominating set of size k in G in the given instance of Dominating set, C_A can change the single initial dud robot in T first to the vertex robots corresponding to the vertices in that dominating set and then the checker robot. This will ensure that the encoding of that k-dominating set is correctly written in the northmost k squares of the sole column of E and then verified to allow the construction of X at p_X. As at most one transition is enabled at any time in the robot in T, the operation of T in E is deterministic. As for (c_1, c_2)-completability, observe that X is created at p_X in exactly $t_{\max} = k + 1$ timesteps. As $k + 1 = |E| < c_1(|E| + |Q|)^{c_2}$ when $c_1 = c_2 = 1$, this means that the construction task is (c_1, c_2)-completable when $c_1 = c_2 = 1$. Conversely, if there is a set C_A of robot-type changes to T that allows the creation of X at p_X, the dud robot must first have been changed to k vertex robots (to both create an encoding of a candidate dominating set of size at most k in G in the given instance

of Dominating set in the northmost k squares in the sole column of E and move the robot to $E_{1,1}$) and then changed to the checker robot (to verify that the encoding is in fact a dominating set of size at most k in G and then construct X at p_X). This establishes the correctness of the reduction and completes the proof. □

Lemma A.11. Dominating set *polynomial-time reduces to* $SelAlg_D$ *such that in the constructed instance of* $SelAlg_D$, $|T| = h = |L| = |X| = c_1 = c_2 = 1$, $|f| = 3$, $|E_T| = 4$, *and* k_a *are functions of* k *in the given instance of* Dominating set.

Proof. Consider the following reduction from Dominating set to SelAlg_D. Given an instance $\langle G = (V, E), k\rangle$ of Dominating set, construct an instance $\langle E, E_T, T, L, p_I, X, p_X, t_{\max}, k_a\rangle$ of SelAlg_D as follows: Let E be a $1 \times |V| + 2$ grid in which $E_{1,1}$ has type e_{F0} and all other squares have type e_0, $E_T = \{e_0, e_1, e_{F0}, e_X\}$, and L consist of a single robot *rob* based on states $Q = \{q_0, q_1, \ldots, q_{(|V|+1)}\}$ with sensory-radius $r = |V| + 1$ and the following transitions:

(1) $\langle q_0, enval(e_0, (0, 0)), enmod(e_1, (0, 0)), goSouth, q_1\rangle$
(2) $\langle q_1, enval(e_0, (0, 0)), *, goSouth, q_1\rangle$
(3) $\{\langle q_i, enval(e_{F0}, (0, 0))$ and $\text{enval}(\text{e}_1, (0, \text{j})), *, \text{stay}, \text{q}_{\text{i+1}}\rangle \mid 1 \le i < |V|$ and $\text{v}_\text{j} \in \text{N}_\text{C}(\text{v}_\text{i})\} \cup$
$\{\langle q_{|V|}, enval(e_{F0}, (0, 0))$ and $\text{enval}(\text{e}_1, (0, \text{j})), \text{enmod}(\text{e}_\text{X}, (0, 0)), \text{stay}, q_{(|V|+1)}\rangle \mid v_j \in N_C(v_{|V|})\}$

Finally, let T consist of *rob* positioned initially at $p_I = E_{1,|V|+2}$, X be a single square of types e_X at position $p_X = E_{1,1}$, $t_{\max} = |V| + 1$, and $k_a = k$. This construction can be done in time polynomial in the given instance of Dominating set.

We now need to verify the correctness of this reduction:

(1) Suppose there a dominating set of size k in G in the given instance of Dominating set; let $V' \subseteq V$ be the vertices in this dominating set, $seq_{V'}$ be the sequence of indices of the vertices in V' sorted in descending order, and $seq_{V'}(i)$, $1 \le i \le k$, be the ith element in that sequence. Let C_A be the sequence of controller-changes that effectively restarts *rob* in q_0 at timesteps in the set $\{(|V| - seq_{V'}(i)) + 2 \mid 1 \le i \le k\}$. Observe that C_A will both force *rob* to encode V' in the middle $|V|$ squares in the sole column in E and position *rob* in state q_1 at $E_{1,1}$, where the transitions in set (3) above will both verify that the dominating set encoded in E is an actual dominating set in G and construct X at p_X. As the set of transitions enabled at any point in the operation of *rob* all do the same thing, the operation of T in E is deterministic. As for (c_1, c_2)-completability, observe that *rob* requires $|V| + 1$ timesteps to reach $E_{1,1}$ and then a further $|V|$ timesteps to create X at p_X. As $2|V| + 1 < ((|V| + 2) + (|V| + 2)) = c_1(|E| + |Q|)^{c_2}$ when $c_1 = c_2 = 1$, the construction task is (c_1, c_2)-completable when $c_1 = c_2 = 1$.
(2) Conversely, suppose there is a set of controller-changes C_A that allows *rob* started at p_I to create X at p_X. By the structure of the transitions in set (3) above, C_A must have made *rob* change a set of squares in the middle $|V|$ squares in the sole column in E to encode a dominating set in G; as such changes can only occur once each time *rob*'s controller is reset and $|C_A| \le k_a = k$, this encoded dominating set can have at most k vertices.

This establishes the correctness of the reduction and completes the proof. □

Lemma A.12. Dominating set *polynomial-time reduces to* $SelAlg_D$ *such that in the constructed instance of* $SelAlg_D$, $|T| = h = |L| = |X| = c_1 = c_2 = 1$, $|Q| = d = 2$, $|E_T| = 4$, *and* k_a *is a function of* k *in the given instance of* Dominating set.

Proof. Modify the reduction in the proof of Lemma A.11 such that *rob* is based on states q_0 and q_1 and the transitions in set (3) are replaced by a single transition of the form $\langle q_1, enval(e_{F0}, (0,0))$ and $(\mathbf{F})$, $enmod(e_X, (0,0)), stay, q_1\rangle$, where $\mathbf{F}$ is the conjunction of $|V|$ parenthesis-enclosed formula, the ith of which for $1 \leq i \leq |V|$ is the disjunction of the predicates in the set $\{enval(e_1, (0, j)) \mid v_j \in N_C(v_i)\}$. The proof of correctness of this modified reduction is essentially the same as that for the reduction in the proof of Lemma A.11, modulo the observation that the construction of X at p_X now requires a total of only $|V| + 2$ timesteps. □

Lemma A.13. *3SAT polynomial-time reduces to SelEnvInf$_D$ such that in the constructed instance of SelEnvInf$_D$, $|T| = h = |X| = t_{\max} = c_1 = 1$, $d = 2$, $c_2 = 3$, and $|f| = |E_T| = 5$.*

Proof. Adapted from the proof of Lemma 6 in the Supplementary Materials for [62]. In that proof, for a given instance of 3SAT, we construct an environment E specified in a $2 \times (|U| + 1)$ grid such that $E_{2,1}$ has type e_{F1}, the northmost $|U|$ squares in the westmost column either have type e_F or e_T, and all other squares have type e_0. A given robot *rob* initially placed at $E_{1,1}$ is structured such that *rob* can only progress to $p_X = E_{2,1}$ to create the single-square structure X if the northmost $|U|$ squares in the westmost column of E specify a satisfying truth assignment for C in the given instance of 3SAT.

In this proof, we have a single-robot swarm T consisting of *rob*, construct an E in which $E_{2,1} = e_{F1}$ and all other squares have type e_0, and set $t_{max} = 1$ and $k_e = |U|$. This construction can be done in time polynomial in the size of the given instance of 3SAT. If there is a satisfying truth assignment for C to the variables in U in the given instance of 3SAT, C_E can change the types of the northmost $|U|$ squares in the westmost column of E to correspond to that truth assignment. Conversely, if there is a set C_E of environment square-type changes that allows *rob* to create X, the northmost $|U|$ squares in the westmost column of E must have had their types changed to correspond to a satisfying truth assignment for C to the variables in U in the given instance of 3SAT. This establishes the correctness of the reduction and completes the proof. □

Lemma A.14. *3SAT polynomial-time reduces to SelEnvInf$_D$ such that in the constructed instance of SelEnvInf$_D$, $|T| = h = |X| = |Q| = t_{\max} = c_1 = c_2 = 1$ $d = 2$, and $|E_T| = 5$.*

Proof. Adapted from the proof of Lemma 7 in the Supplementary Materials for [62], In that proof, the robot *rob* is modified from that in the proof of Lemma 6 in the Supplementary Materials of [62] to have a single transition whose (now rather long) activation formula checks if the northmost $|U|$ squares in the westmost column of E specify a satisfying truth assignment for C to the variables in U in the given instance of 3SAT. In this proof proof, we construct T and E along the lines in the proof of Lemma A.13 above. □

Lemma A.15. Clique *polynomial-time reduces to SelEnvInf$_D$ such that in the constructed instance of SelEnvInf$_D$, $|T| = h = |X| = c_1 = t_{\max} = 1$, $c_2 = 2$, $|f| = 3$, and $|Q|$, r, $|E|$, and k_e are functions of k in the given instance of* Clique.

Proof. Adapted from the proof of Lemma 8 in the Supplementary Materials for [62]. In that proof, for a given instance of Clique, we construct an environment E specified in a $2 \times (|k| + 1)$ grid such that $E_{2,1}$ has type e_{F1}, the northmost k squares in the westmost column either have types from the set $\{e_1, e_2, \ldots, e_{|V|}\}$, and all other squares have type e_0. A given robot *rob* initially placed at $E_{1,1}$ is structured such that *rob* can only progress to $p_X = E_{2,1}$ to create the single-square structure

X if the northmost k squares in the westmost column of E specify a clique of size k in G in the given instance of Clique.

In this proof, we have a single-robot swarm T consisting of rob, construct an E in which $E_{2,1} = e_{F1}$ and all other squares have type e_0, and set $t_{max} = 1$ and $k_e = k$. This construction can be done in time polynomial in the size of the given instance of Clique. If there is a clique of size k in G in the given instance of Clique, C_E can change the types of the northmost k squares in the westmost column of E to correspond to that clique. Conversely, if there is a set C_E of environment square-type changes that allows rob to create X, the northmost k squares in the westmost column of E must have had their types changed to correspond to a clique of size k in G in the given instance of Clique. This establishes the correctness of the reduction and completes the proof. □

Lemma A.16. Clique *polynomial-time reduces to SelEnvInf$_D$ such that in the constructed instance of SelEnvInf$_D$, $|T| = h = |X| = |Q| = t_{\max} = c_1 = c_2 = 1$, $d = 2$, and r, $|E|$, and k_e are functions of k in the given instance of* Clique.

Proof. Adapted from the proof of Lemma 9 in the Supplementary Materials for [62], In that proof, the robot rob is modified from that in the proof of Lemma 8 in the Supplementary Materials of [62] to have a single transition whose (now rather long) activation formula checks if the northmost k squares in the westmost column of E specify a clique of size k in G in the given instance of Clique. In our proof proof, we construct T and E along the lines in the proof of Lemma A.15 above. □

Lemma A.17. Dominating set *polynomial-time reduces to SelLead$_D$ such that in the constructed instance of SelLead$_D$, $|Q| = c_1 = c_2 = 1$, $d = 3$, $|E_T| = 5$, $h = |X| = t_{\max} = 1$, and k_l is a function of k in the given instance of* Dominating set.

Proof. Given an instance $\langle G, k\rangle$ of Dominating set, construct an instance $\langle E, E_T, T, p_I, X, p_X, t_{\max}, k_l\rangle$ of SelLead$_D$ as follows: Let $E_T = \{e_D, e_C, e_{C1}, e_{C2}, e_X\}$ and E be based on a $|V| \times 4$ grid such that the square types of $E_{1,3}$ and $E_{1,4}$ are E_{C1} and E_{C2}, respectively, and all others squares have type e_D. Let T consist of $|V| + 1$ copies of robot r_G, each of which has a single state q_0, sensory radius $r = |V|$, and the following transitions:

(1) $\langle q_0, enval(e_{C1}, (0,0))$ and $\mathbf{F}, enval(e_C, (0,0)), goNorth, q_0\rangle$, where $\mathbf{F}$ is the conjunction of $|V|$ clauses, the ith of which is a parenthesis-enclosed disjunction of the set of expressions $\{enval(e_{robot}, (i,2)) \mid v_i \in N_C(v)\}$.
(2) $\langle q_0, enval(e_C, (0,-1))$ and $enval(q_{C2}, (0,0)), enmod(e_X, (0,0)), stay, q_0\rangle$
(3) $\langle q_0, (enval(e_D, (0,0))$ or $enval(e_X, (0,0))), *, stay, q_0\rangle$

Let p_I be such that $|V|$ copies of r_G are in the southmost row of E and the $(|V| + 1)$st copy is at $E_{1,3}$; call the first set of robots variable-robots and the final robot the checker-robot. Let X be a single square of type e_X at $p_X = E_{1,4}$. Finally, set $t_{\max} = 1$ and $k_l = k$. Note that this instance of SelLead$_D$ can be constructed in time polynomial in the size of the given instance ofDominating set.

We now need to verify the correctness of this reduction:

- Suppose there is a dominating set of size k in G in the given instance of Dominating set. Let $D \subseteq V$ be the vertices in this dominating set and C_L be the set of position-changes that shifts all of the variable-robots in the southmost row of E corresponding to vertices in D one square north. By the structure of r_G, the positions of the moved variable-robots will allow $\mathbf{F}$ in the activation-formula of transition (1) of the checker-robot to evaluate to True, which will allow the checker-robot to proceed northwards to construct X at p_X. Moreover, as the

number of timesteps the checker-robot needs to construct X at p_X is $2 \leq c_1|E|^{c_2} = |E|$, this construction task is (c_1, c_2)-completable when $c_1 = c_2 = 1$.

- Conversely, suppose there is a set of position-changes C_L such that for T started at p_I and modified by C_L, the task of constructing X at p_X is (c_1, c_2)-completable. By the structure of r_G, the checker robot can only construct X at p_X if it had previously changed the type of $E_{1,3}$ to E_C, and that can only happen if the position-changes in C_L shifted variable-robots in the southmost row corresponding to a dominating set for G in the given instance of Dominating set; moreover, as $k_l = k$, this dominating set must be of size k.

This establishes the correctness of the reduction and completes the proof. □

Lemma A.18. Dominating set *polynomial-time reduces to SelLead$_D$ such that in the constructed instance of SelLead$_D$, $h = |X| = c_1 = t_{\max} = 1$, $c_2 = 2$, $|f| = 3$, $|E_T| = 5$, and k_l is a function of k in the given instance of* Dominating set.

Proof. Replace r_G in the proof of Lemma A.17 with a $(|V|^2 + 1)$-state robot based on the states $\{q_0 = q_{N1,1}, q_{N1,2}, \ldots, q_{N1,|V|}, q_{N2,1}, q_{N2,2}, \ldots, q_{N2,|V|}, \ldots, q_{N|V|,1}, q_{N|V|,2}, \ldots, q_{N|V|,|V|}, q_{N(|V|+1),1}\}$ with sensory radius $r = |V|$ and the following transitions:

(1) $|V|$ sets of transitions, where for $1 \leq i \leq |V|$, the ith set consists of the transitions $\{\langle q_{Ni,j}, *, *, stay, q_{Ni,j+1} \mid 1 \leq j \leq |V| - 1\}$ and $\{q_{Ni,j}, enval(e_{robot}, (j, 2)), *, stay, q_{N(i+1),1}\rangle \mid v_j \in N_C(v_i)\}$.
(2) $\langle q_{N(|V|+1),1}, enval(e_{C1}, (0, 0)), enmod(e_C, (0, 0)), goNorth, q_{N(|V|+1),1}\rangle$
(3) $\langle q_{N(|V|+1),1}, enval(e_C, (0, -1))$ and $enval(q_{C2}, (0, 0)), enmod(e_X, (0, 0)), stay, q_{N(|V|+1),1}\rangle$
(4) $\langle q_{N(|V|+1),1}, (enval(e_D, (0, 0))$ or $enval(e_X, (0, 0))), *, stay, q_{N(|V|+1),1}\rangle$

Once again, this construction can be done in time polynomial in the given instance of Dominating set. The proof of correctness of this reductions is that given in the proof of Lemma A.17 modified modulo the following two observations:

(1) The transitions in (1) and (2) above simulate transition (1) in the robot r_G in the proof of Lemma A.17; and
(2) As these modifications to r_G now mean that the number of timesteps the checker-robot needs to construct X at p_X is $\leq |V|^2 + 2 \leq 3|V|^2 \leq 16|V|^2 = (4|V|)^2 = c_1|E|^{c_2}$, this construction task is (c_1, c_2)-completable when $c = 1$ and $c_2 = 2$.

This completes the proof. □

Lemma A.19. Dominating set^{PD3} *polynomial-time reduces to SelLead$_D$ such that in the constructed instance of SelLead$_D$, $h = |X| = c_1 = t_{\max} = 1$, $c_2 = 2$, $d = 4$, $|f| = 3$, and $|E_T| = 5$.*

Proof. Apply reduction in the proof of Lemma A.18 relative to Dominating set^{PD3} instead of Dominating set. □

Lemma A.20. Clique *polynomial-time reduces to SelLead$_D$ such that in the constructed instance of SelLead$_D$, $|T| = h = |X| = c_1 = 1$, $c_2 = 2$, $|f| = 3$, and r, $|E|$, and k_l are functions of k in the given instance of* Clique.

Proof. Inspired by the proof of Lemma 8 in the Supplementary Materials for [62]. In that proof, for a given instance of Clique, we construct an environment E specified in a $2 \times (|k| + 1)$ grid such that $E_{2,1}$ has type e_{F1}, the northmost k squares in the westmost column have types from the set $\{e_1, e_2, \ldots, e_{|V|}\}$, and all other squares have type e_0. A given robot *rob* initially placed at $E_{1,1}$ is structured such that *rob* can only progress to $p_X = E_{2,1}$ to create the single-square structure X if the northmost k squares in the westmost column of E specify a clique of size k in G in the given instance of Clique.

In this proof, we have a swarm T consisting of a single robot that modifies *rob* by having a new initial state $q_0 = q_{W1}$, additional states $\{q_{W2}, \ldots, q_{W|V|}\}$, and the following additional transitions:

(1) $\{\langle q_{Wi}, enval(e_0, (0,0)), enmod(e_{v_i}, (0,0)), stay, q_{W(i+1)}\rangle \mid 1 \leq i \leq |V| - 1\}$; and
(2) $\{\langle q_{Wi}, enval(e_{F0}, (0,0), enmod(e_0, (0,0)), stay, q'_0\rangle \mid 0 \leq i \leq |V|\}$, where q'_0 is the initial state of *rob*.

Let E be such that $E_{1,1} = e_{F0}$ and $E_{2,1} = e_{F1}$ and all other squares have type e_0, $p_I = E_{k+1,1}$, $t_{max} = |V| + 1$, and $k_l = k + 1$. This construction can be done in time polynomial in the size of the given instance of Clique. If there is a clique of size k in G in the given instance of Clique, C_L can change the positions of the robot one square south at the appropriate times during the first $|V| + 1$ timesteps such that first the square-types corresponding to the vertices in that clique are written in the northmost k squares of the first column in E and then the robot is positioned at $E_{1,1}$ to verify this clique and construct X at $E_{2,1}$. Conversely, if there is a set C_L of position-changes to the robot in T that allows the robot to create X, the robot must first have been moved one square south k times at the timesteps corresponding to the vertices in a candidate k-clique (to both create an encoding of that candidate k-clique in G in the given instance of Clique in the northmost k squares in the westmost column of E and move the robot to $E_{1,1}$) and then moved south one final time (to position the robot at $E_{1,1}$ to verify that the encoding is in fact a k-clique in G and then construct X at $E_{2,1}$). This establishes the correctness of the reduction and completes the proof. □

Result A.1: SelAlg is polynomial-time tractable when $h = t_{\max} = 1$.

Proof. The algorithm in the proof of Result A in [60], which tests for each controller c in L whether a swarm based entirely on c can construct X at p_X in at most $c_1|E|^{c_2}$ timesteps, operates in polynomial time. We need only modify that algorithm such that prior to evaluating each c in L, we first verify that k_a controller-changes suffice to convert all controllers in T to c (i.e., are there exactly k_a controllers in T that are not c?), which can also be done in polynomial time. □

Result A.2: If SelAlg is polynomial-time tractable when $h = 2$ and $t_{\max} = 1$ or either SelEnvInf or SelLead is polynomial-time tractable when $h = t_{\max} = 1$ then $P = NP$.

Proof. The NP-hardness of SelAlg_D, SelEnvInf_D, and SelLead_D under the specified values of h and $t_{\max}$ follows from the NP-hardness of Dominating set and 3SAT and the reductions in Lemmas A.7, A.13, and A.17, respectively. The result then follows from Lemma A.1. □

Result A.3: If SelAlg when $h = 2$ and $t_{\max} = 1$ or SelEnvInf or SelLead when $h = t_{\max} = 1$ are polynomial-time promise solvable then $P = NP$.

Proof. Suppose that SelAlg is polynomial-time promise solvable by an algorithm A when $h = 2$ and $t_{\max} = 1$.[16] Consider the following algorithm for 3SAT:

[16] It may initially seem puzzling why we here directly evaluate the polynomial-time promise solvability of SelAlg. This is necessary because the promise solvability of any decision problem such as SelAlg_D is established by the trivial constant-time algorithm which always answers "Yes" (and hence is always correct if a solution exists).

(1) Given an instance I of 3SAT, construct an instance I' of SelAlg using the reduction from 3SAT to SelAlg_D described in Lemma A.7.
(2) Run A on I' to produce output O' for SelAlg.
(3) As specified in the converse part of the proof of correctness of the reduction in Lemma A.7 use the types of the C_A-changed robots in O' to derive a candidate truth assignment O for the given instance of 3SAT.
(4) If O is a correct solution for I, output "Yes"; otherwise, output "No" (as by the definition of promise solvability, if the answer was "Yes" then A would have had to output O' such that O was a correct solution to the given instance of 3SAT).

As all steps in this algorithm run in polynomial time, the above is a polynomial-time algorithm for 3SAT. However, given the NP-hardness of 3SAT, this would imply that $P = NP$, completing the proof of polynomial-time promise non-solvability for SelAlg. Similar arguments using the reductions from 3SAT and Dominating set to SelEnvInf_D and SelLead_D in Lemmas A.13 and A.17 along with the NP-hardness of 3SAT and Dominating set establish the polynomial-time promise non-solvability of SelEnvInf and SelLead. □

Result A.4: If SelAlg when $h = 2$ and $t_{\max} = 1$ or SelEnvInf or SelLead when $h = t_{\max} = 1$ are solvable by polynomial-time algorithms with polynomial error frequencies (i.e., $err(n)$ is upper bounded by a polynomial of n) then $P = NP$.

Proof. That polynomial-time promise solvability for any of these problems under the stated restriction implies $P = NP$ follows from the NP-hardness of each of these problems (which is established in the proof of Result A.2) and Corollary 2.2 in [22]. □

Result A.5: if $P = BPP$ and SelAlg when $h = 2$ and $t_{\max} = 1$ or SelEnvInf or SelLead when $h = t_{\max} = 1$ are polynomial-time solvable by probabilistic algorithms which operate correctly with probability $\geq 2/3$ then $P = NP$.

Proof. It is widely believed that $P = BPP$ [68, Section 5.2] where BPP is considered the most inclusive class of decision problems that can be efficiently solved using probabilistic methods (in particular, methods whose probability of correctness is $\geq 2/3$ and can thus be efficiently boosted to be arbitrarily close to one). Hence, if any of SelAlg_D, SelEnvInf_D, or SelLead_D under the stated restrictions has a probabilistic polynomial-time algorithm which operates correctly with probability $\geq 2/3$ then that problem is by definition in BPP. However, if $BPP = P$ and we know that each of these problems is NP-hard by the proof of Result A.2, this would then imply by the definition of NP-hardness that $P = NP$, completing the proof. □

Result B.SA.1: If $\langle |T|, h, |Q|, |f|, r, |X|, t_{\max}, k_a \rangle$-SelAlg is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[2]$-hardness of k-Dominating set, the inclusion of $W[1]$ in $W[2]$, the reduction in Lemma A.5, and Lemma A.2. □

Result B.SA.2: If $\langle |T|, h, |Q|, |E_T|, |X|, t_{\max}, k_a \rangle$-SelAlg is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[2]$-hardness of k-Dominating set, the inclusion of $W[1]$ in $W[2]$, the reduction in Lemma A.6, and Lemma A.2. □

Result B.SA.3: If $\langle h, |Q|, d, |f|, r, |E_T|, |L|, t_{\max}, k_a \rangle$-SelAlg is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[1]$-hardness of k-W3SAT, the reduction in Lemma A.8, and Lemma A.2. □

Result B.SA.4: If $\langle |T|, h, |Q|, |f|, r, |E|, |X|, t_{\max}, k_a \rangle$-SelAlg is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[1]$-hardness of k-Clique, the reduction in Lemma A.9, and Lemma A.2. □

Result B.SA.5: If $\langle |T|, h, |Q|, d, r, |E|, |X|, t_{\max}, k_a \rangle$-SelAlg is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[1]$-hardness of k-Dominating set, the reduction in Lemma A.10, and Lemma A.2. □

Result B.SA.6: If $\langle |T|, h, |f|, |L|, |E_T|, |X|, k_a \rangle$-SelAlg is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[1]$-hardness of k-Dominating set, the reduction in Lemma A.11, and Lemma A.2. □

Result B.SA.7: If $\langle |T|, h, |Q|, d, |L|, |E_T|, |X|, k_a \rangle$-SelAlg is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[1]$-hardness of k-Dominating set, the reduction in Lemma A.12, and Lemma A.2. □

Result B.SA.8: $\langle |T|, |L|, t_{\max} \rangle$-SelAlg is fp-tractable.

Proof. Consider the algorithm that selects all possible combinations of k_a elements from the set of valid timesteps for changes, members of T, and possible changes from L to members of T, and for each such combination, verifies whether or not T started at p_I and subsequently modified by the changes in that combination constructs X at p_X. As this verification can be done in time polynomial in the input size, the running time of this algorithm is upper bounded by $(t_{\max}|T||L|)^{k_a}$ times some polynomial of the input size. To get the result. observe that $k_a \leq |T| \times t_{\max}$, as the maximum possible value of k_a occurs when every robot in T has its controller changed in every timestep up to and including timestep t_{max}. □

Result B.SA.9: $\langle |T|, |Q|, r, |E_T|, t_{\max} \rangle$-SelAlg is fp-tractable.

Proof. Follows from the algorithm in the proof of Result B.SA.8 and the proof of Result I in [60], in which it is shown that $|L|$ is upper-bounded by a function of $|Q|$, r, and $|E_T|$. □

Result B.SE.1: If $\langle |T|, h, d, |f|, |E_T|, |X|, t_{\max} \rangle$-SelEnvInf is fp-tractable then $P = NP$.

Proof. The NP-hardness of SelEnvInf$_D$ when $|T| = h = |X| = t_{\max} = 1, d = 2$, and $|f| = |E_T| = 5$ follows from the NP-hardness of 3SAT and the reduction in Lemma A.13. The result then follows from Lemma A.4. □

Result B.SE.2: If $\langle |T|, h, |Q|, d, |E_T|, |X|, t_{\max} \rangle$-SelEnvInf is fp-tractable then $P = NP$

Proof. The NP-hardness of SelEnvInf$_D$ when $|T| = h = |X| = |Q| = t_{\max} = 1$, $d = 2$, and $|E_T| = 5$ follows from the NP-hardness of 3SAT and the reduction in Lemma A.14. The result then follows from Lemma A.4. □

Result B.SE.3: If $\langle |T|, h, |Q|, |f|, r, |E|, |X|, t_{\max}, k_e \rangle$-SelEnvInf is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[1]$-hardness of $\langle k \rangle$-Clique, the reduction in Lemma A.15, and Lemma A.2. □

Result B.SE.4: If $\langle |T|, h, |Q|, d, r, |E|, |X|, t_{\max}, k_e \rangle$-SelEnvInf is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[1]$-hardness of $\langle k \rangle$-Clique, the reduction in Lemma A.16, and the definition of FPT. □

Result B.SE.5: $\langle |E|, |E_T|, t_{\max} \rangle$-SelEnvInf is fp-tractable.

Proof. Consider the algorithm that selects all possible combinations of k_e elements from the set of valid timesteps for changes, squares in E, and possible changes from E_T to squares in E, and for each such combination, verifies whether or not T started at p_I in E and subsequently modified by the changes in that combination constructs X at p_X. As this verification can be done in time polynomial in the input size, the running time of this algorithm is upper bounded by $(t_{\max}|E||E_T|)^{k_e}$ times some polynomial of the input size. To get the result, observe that $k_e \leq |E| \times t_{\max}$, as the maximum possible value of k_e occurs when every square in E has its type changed in every timestep up to and including timestep t_{max}. □

Result B.SL.1: If $\langle h, |Q|, d, |E_T|, |X|, t_{\max}, k_l\rangle$-SelLead is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[2]$-hardness of k-Dominating set, the inclusion of $W[1]$ in $W[2]$, the reduction in Lemma A.17, and Lemma A.2. □

Result B.SL.2: If $\langle h, |f|, |E_T|, |X|, t_{\max}, k_l\rangle$-SelLead is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[2]$-hardness of k-Dominating set, the inclusion of $W[1]$ in $W[2]$, the reduction in Lemma A.18, and Lemma A.2. □

Result B.SL.3: If $\langle h, d, |f|, |E_T|, |X|, t_{\max}\rangle$-SelLead is fp-tractable then $P = NP$

Proof. The NP-hardness of SelLead$_D$ when $|f| = t_{\max} = 1$, $h = 2$, $d = 4$, $|f| = 3$, $|E_T| = 5$, and $|X| = 1$ follows from the NP-hardness of Dominating set^{PD3} and the reduction in Lemma A.19. The result then follows from Lemma A.4. □

Result B.SL.4: If $\langle |T|, h, |f|, r, |E|, |X|, k_l\rangle$-SelLead is fp-tractable then $FPT = W[1]$.

Proof. Follows from the $W[1]$-hardness of k-Clique, the reduction in Lemma A.20, and Lemma A.2. □

Result B.SL.5: $\langle |T|, t_{\max}\rangle$-SelLead is fp-tractable.

Proof. Consider the algorithm that selects all possible combinations of k_l elements from the set of valid timesteps for changes, members of T, and possible position changes to members of T, and for each such combination, verifies whether or not T started at p_I and subsequently modified by the changes in that combination constructs X at p_X. As this verification can be done in time polynomial in the input size, the running time of this algorithm is upper bounded by $(t_{\max}|T|4)^{k_l}$ times some polynomial of the input size. To get the result, observe that $k_l \leq |T| \times t_{\max}$, as the maximum possible value of k_l occurs when every robot in T has its position changed in every timestep up to and including timestep t_{max}. □

This useful observation follows from the definition of $|x|^k$-tractability.

Observation 1. *Given a problem* **X** *and an aspect-set* K *for* **X**, *any* $|x|^k$*-tractable algorithm* A *for* $\langle K'\rangle$-**X**, $K' \subseteq K$, *is also a polynomial-time algorithm for* **X** *when all aspects of* K' *have constant value.*

Results C.SA.1–C.SA.5, C.SE.1, C.SE.2, and C.SL.1–C.SL.3 in turn follow from this observation and the reductions in the proofs of Results B.SA.3–B.SA.7, B.SE.1, B.SE.2, and B.SL.1, B.SL.3, and B.SL.4, respectively, which show the NP-hardness of their associated problems when the listed aspects are of constant value.

Result C.SA.1: If $\langle h, |Q|, d, |f|, r, |E_T|, |L|, t_{\max}\rangle$-SelAlg is $|x|^k$-tractable then $P = NP$.
Result C.SA.2: If $\langle |T|, h, |f|, |X|\rangle$-SelAlg is $|x|^k$-tractable then $P = NP$.
Result C.SA.3: If $\langle |T|, h, |Q|, d, |X|\rangle$-SelAlg is $|x|^k$-tractable then $P = NP$.

Result C.SA.4: If $\langle |T|, h, |f|, |E_T|, |X|, |L| \rangle$-SelAlg is $|x|^k$-tractable then $P = NP$.
Result C.SA.5: If $\langle |T|, h, |Q|, d, |E_T|, |X|, |L| \rangle$-SelAlg is $|x|^k$-tractable then $P = NP$.
Result C.SA.6: $\langle k_a \rangle$-SelAlg is $|x|^k$-tractable.

Proof. Follows from the algorithm in the proof of Result B.SA.8. □

Result C.SE.1: If $\langle |T|, h, d, |f|, |E_T|, |X|, t_{\max} \rangle$-SelEnvInf is $|x|^k$-tractable then $P = NP$.
Result C.SE.2: If $\langle |T|, h, |Q|, d, |E_T|, |X|, t_{\max} \rangle$-SelEnvInf is $|x|^k$-tractable then $P = NP$.
Result C.SE.3: $\langle k_e \rangle$-SelEnvInf is $|x|^k$-tractable.

Proof. Follows from the algorithm in the proof of Result B.SE.5. □

Result C.SL.1: If $\langle h, |Q|, d, |E_T|, |X|, t_{\max} \rangle$-SelLead is $|x|^k$-tractable then $P = NP$.
Result C.SL.2: If $\langle h, d, |f|, |E_T|, |X|, t_{\max} \rangle$-SelLead is $|x|^k$-tractable then $P = NP$.
Result C.SL.3: If $\langle |T|, h, |f|, |X| \rangle$-SelLead is $|x|^k$-tractable then $P = NP$.
Result C.SL.4: $\langle k_l \rangle$-SelLead is $|x|^k$-tractable.

Proof. Follows from the algorithm in the proof of Result B.SL.5. □

Consider the following restricted form of parameterized reducibility.

Definition A.21. (Adapted from the definition on page 1359 of [6]) A problem Π with parameter k is linearly fpt-reducible to a problem Π' with parameter k' if there exists an algorithm A with running time $f(k)m^{o(k)}|x|^c$ for some computable function $f()$ and constant c such that for each instance x of Π with search space and instance sizes m and $|x|$, the algorithm A produces an instance x' of Π' such that $m' \leq c_1 m^{c_2}$ and $|x'| \leq c_3|x|^{c_4}$ for positive constants c_1, c_2, c_3, and c_4, $k' \leq c_5 k$ for positive constant c_5, and x is a yes-instance of Π if and only if x' is a yes-instance of Π'.

Result C.SA.7: If $\langle k_a \rangle$-SelAlg is $|x|^k$-tractable such that $f(k_a) = o(k_a)$ then the Exponential Time Hypothesis is false.

Proof. Consider the reduction in the proof of Lemma A.5 which creates an instance of SelAlg_D that is of size polynomial in the size of the given instance of Dominating set. Moreover, as $|T| = |V| + 1$, $|L| = |V| + 2$, $t_{\max} = 1$, and $k_a = k + 1$ in each created instance of SelAlg_D, by the algorithm for SelAlg in the proof of Result B.SA.8, this instance is solvable by selecting $k' = k_a = k + 1 \leq 2k$ elements from a search space of size

$$\begin{aligned} m' &= |T||L| \\ &= (|V| + 1)(|V| + 2) \\ &= |V|^2 + 3|V| + 2 \\ &\leq |V|^2 + 3|V|^2 + 2|V|^2 \\ &= 6|V|^2 \end{aligned}$$

(where this search space corresponds to the choices of elements in T to replace with elements from L and those choices of elements from L). Problem $\langle k \rangle$-Dominating set has a search space of size $m = |V|$ as it can be solved by an algorithm that generates all $|V|^k$ k-vertex subsets of the given graph G and checks each subset to see if it is a dominating set in G. This establishes that the reduction in the proof of Lemma A.5 is a linear fpt reduction from $\langle k \rangle$-Dominating set to $\langle k_a \rangle$-SelAlg_D. As $\langle k \rangle$-Dominating set is $W_l[2]$-hard [6, Theorem 5.4], $\langle k_a \rangle$-SelAlg_D is $W_l[2]$-hard. Moreover, as $W_l[1] \subseteq W_l[2]$, by the definition of $W_l[2]$-hardness, every problem in $W_l[1]$ is also linearly fpt-reducible to $\langle k_a \rangle$-SelAlg_D and $\langle k_a \rangle$-SelAlg_D is $W_l[1]$-hard as well. However, no $W_l[1]$-hard problem $\langle k \rangle$-**X** is solvable in $h(k)|x|^{o(k)}$ time for some function $h()$ unless the Exponential

Time Hypothesis is false [6, Theorem 5.8]. As this also rules out an algorithm with running time $|x|^{o(k)}$, $\langle k_a \rangle$-SelAlg$_D$ (and hence $\langle k_a \rangle$-SelAlg by Lemma A.3) cannot be $|x|^k$-tractable such that $f(k_a) = o(k_a)$ unless the Exponential Time Hypothesis is false, completing the proof. □

Result C.SE.4: If $\langle k_e \rangle$-SelEnvInf is $|x|^k$-tractable such that $f(k_e) = o(k_e)$ then the Exponential Time Hypothesis is false.

Proof. Consider the reduction in the proof of Lemma A.15 which creates an instance of SelEnvInf$_D$ that is of size polynomial in the size of the given instance of Clique. Moreover, as $|E| = 2k + 2$, $|E_T| = |V| + 2$, $t_{max} = 1$, and $k_e = k$ in each created instance of SelEnvInf$_D$, by the algorithm for SelEnvInf in the proof of Result B.SE.5, this instance is solvable by selecting $k' = k_e = k$ elements from a search space of size

$$\begin{aligned} m' &= |E||E_T| \\ &= (2k+2)(|V|+2) \\ &\leq (2|V|+2)(|V|+2) \\ &= 2|V|^2 + 6|V| + 4 \\ &\leq 2|V|^2 + 6|V|^2 + 4|V|^2 \\ &= 12|V|^2 \end{aligned}$$

(where this search space corresponds to the choices of elements in E to replace with elements from E_T and those choices of elements from E_T). Problem $\langle k \rangle$-Clique has a search space of size $m = |V|$ as it can be solved by an algorithm that generates all $|V|^k$ k-vertex subsets of the given graph G and checks each subset to see if it is a clique in G. This establishes that the reduction in the proof of Lemma A.5 is a linear fpt reduction from $\langle k \rangle$-Clique to $\langle k_e \rangle$-SelEnvInf$_D$. As $\langle k \rangle$-Clique is $W_l[1]$-hard [6, Theorem 5.5], $\langle k_e \rangle$-SelEnvInf$_D$ is also $W_l[1]$-hard. However, no $W_l[1]$-hard problem $\langle k \rangle$-**X** is solvable in $h(k)|x|^{o(k)}$ time for some function $h()$ unless the Exponential Time Hypothesis is false [6, Theorem 5.8]. As this also rules out an algorithm with running time $|x|^{o(k)}$, $\langle k_e \rangle$-SelEnvInf$_D$ (and hence $\langle k_e \rangle$-SelEnvInf by Lemma A.3) cannot be $|x|^k$-tractable such that $f(k_a) = o(k_e)$ unless the Exponential Time Hypothesis is false, completing the proof. □

Result C.SL.5: If $\langle k_l \rangle$-SelLead is $|x|^k$-tractable such that $f(k_l) = o(k_l)$ then the Exponential Time Hypothesis is false.

Proof. Consider the reduction in the proof of Lemma A.17 which creates an instance of SelLead$_D$ that is of size polynomial in the size of the given instance of Dominating set. Moreover, as $|T| = |V| + 1$, $t_{\max} = 1$, and $k_l = k$ in each created instance of SelLead$_D$, by the algorithm for SelLead in the proof of Result B.SL.5, this instance is solvable by selecting $k' = k_l = k$ elements from a search space of size

$$\begin{aligned} m' &= 4|T| \\ &= 4(|V|+1) \\ &= 4|V| + 4 \\ &\leq 4|V| + 4|V| \\ &= 8|V| \end{aligned}$$

(where this search space corresponds to the choices of elements in T to move and those choices of movements). Problem $\langle k \rangle$-Dominating set has a search space of size $m = |V|$ as it can be solved by an algorithm that generates all $|V|^k$ k-vertex subsets of the given graph G and checks each subset to see if it is a dominating set in G. This establishes that the reduction in the proof of Lemma A.17 is a linear fpt reduction from $\langle k \rangle$-Dominating set to $\langle k_l \rangle$-SelLead$_D$. As $\langle k \rangle$-Dominating set is $W_l[2]$-hard [6, Theorem 5.4], $\langle k_l \rangle$p-SelLead$_D$ is also $W_l[2]$-hard. Moreover, as $W_l[1] \subseteq W_l[2]$, by the definition of $W_l[2]$-hardness, every problem in $W_l[1]$ is also linearly fpt-reducible to $\langle k_l \rangle$-SelLead$_D$ and $\langle k_l \rangle$-SelLead$_D$ is $W_l[1]$-hard as well. However, no $W_l[1]$-hard problem $\langle k \rangle$-**X** is solvable in $h(k)|x|^{o(k)}$ time for some function $h()$ unless the Exponential Time Hypothesis is false [6, Theorem 5.8]. As this also rules out an algorithm with running time $|x|^{o(k)}$, $\langle k_l \rangle$-SelLead$_D$ (and hence $\langle k_l \rangle$-SelLead by Lemma A.3) cannot be $|x|^k$-tractable such that $f(k_a) = o(k_a)$ unless the Exponential Time Hypothesis is false, completing the proof. □