Exploring Algorithmic Options for the Efficient Design and Reconfiguration of Reactive Robot Swarms

Todd Wareham
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL Canada A1B 3X5
harold@mun.ca

ABSTRACT
A key challenge in robot swarm engineering is the design of individual robot controllers such that the robots as a group can perform a specified task. In this paper, we explore algorithmic options for designing and reconfiguring swarms of synchronous reactive robots to perform a joint navigation / morphogenesis task in a known world. Our results show that neither of these problems can be solved both efficiently and correctly either in general or relative to a surprisingly large number of restrictions on robot and swarm architecture. We also give restrictions under which these problems can be solved both efficiently and correctly.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; I.2.9 [**Artificial Intelligence**]: Robotics; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical algorithms and problems—*computations on discrete structures*

General Terms
Algorithms, Design, Measurement, Performance, Theory

Keywords
swarm robotics, subsumption architecture, parameterized computational complexity

1. INTRODUCTION
A robot swarm is a group of autonomous robots that can collaborate without a centralized controller to perform tasks that no individual robot in the swarm can accomplish. These tasks range from specific ongoing behaviours such as the systematic exploration and exploitation of the resources in an unknown environment to the formation of particular structures, either by manipulation of the environment or assembling the robots in the swarm into particular configurations (**morphogenesis**). Swarm and morphogenetic engineering [4, 9] are emerging disciplines whose respective goals are the efficient and correct creation of swarms with specific collective behaviours and structure-formation capabilities.

A key problem in both swarm and morphogenetic engineering is the design of controllers for the individual robots that will allow these robots as a group to perform their assigned task. Two main types of swarm design methodologies have been proposed [4, 9]: behaviour-based design (in which individual robot controllers are incrementally modified (often by hand) such that the robots as a group converge on performing the desired task) and automatic design (in which various search-based methods like evolutionary algorithms or reinforcement learning create the individual robot controllers). To date, there is no generally-applicable method that is guaranteed to design swarms efficiently and reliably for all inputs. Recent work [14] suggests that such methods may be possible under restrictions, but proposes no technique except intuition-guided experiments for finding out what these restrictions are. Given the above, it would be most useful to know if efficient and correct swarm design methods are possible and, if so, under what circumstances.

In this paper, we present initial results addressing both of these questions. First, using computational complexity analysis [15], we show that both designing and reconfiguring swarms of synchronous reactive robots (either by changes to their controllers or initial positions in the environment) to perform a joint navigation / morphogenesis task in a known world is NP-hard and thus intractable in general. Second, using parameterized complexity analysis [10], we establish restrictions on individual robot and swarm architectures that make these problems tractable and prove that tractability holds relative to surprisingly few such restrictions. Though the results above are derived for a particular robot and swarm architecture, we show that they apply to a much broader class of architectures.

The remainder of this paper is organized as follows. In Section 2, we present our robot and swarm architecture and formalize basic swarm design and reconfiguration problems. Section 3 demonstrates the intractability of these problems. Section 4 describes a methodology for identifying conditions for tractability, which is then applied in Section 5 to identify such conditions for our problems. In order to accommodate

BICT 2015, December 03-05, New York City, United States
Copyright © 2016 ICST
DOI 10.4108/eai.3-12-2015.2262395

conference page limits and focus in the main text on the implications of our results for swarm engineering, result proof sketches are given in Appendix A. Finally, our conclusions and directions for future work are given in Section 6.

1.1 Related Work

Various work has been done on the computational complexity of verifying if a given swarm-like system can perform a task and designing such systems for tasks. The systems so treated include groups of agents [11, 20, 24], robots [8, 16], game-pieces [12], and tiles [1]. Much of this work, *e.g.*, [1, 8, 12, 16] assumes that the entities being moved cannot sense, plan, or move autonomously. In the work where entities do have these abilities, *e.g.*, [11, 20, 24], the formalizations of control mechanisms and environments are very general and powerful (*e.g.*, arbitrary Turing machines or Boolean propositional formulae), rendering both the intractability of these problems unsurprising and the derived results unenlightening with respect to possible restrictions that could yield tractability. Only one complexity-theoretic work to date incorporates both autonomous robots and a suitably simple and explicit model of robot architecture and environment [22]. Though this work deals only with single robots, it is the basis for the research presented in this paper.

2. BACKGROUND

2.1 Formalizing Reactive Robots

Our robots will exist within a a finite square-based map W in which basic compass movement is possible between adjacent squares, *i.e.*, north, south, east, and west, and each square is either a freespace (which a robot can occupy or travel through) or an obstacle. Each square has an associated type; let this set of types be denoted by E. Note that these worlds are static, in that the map cannot be changed by the robots.

Our reactive robot will be a simplified Brooks-style architecture [5] consisting of sensors, a set of layers, a total ordering on these layers, and a set of subsumption connections between layers. These components are specified as follows:

- The sensors can see outwards in a radius r around the robot in every direction up to the closest obstacle in that direction, and can verify, for each square-type $e \in E$, the presence of e at any specified position pos within that perceptual radius, *i.e.*, $exists(pos, e)$. Each robot also has a compass that allows it to orient itself relative to the north-south and east-west axes.
- Each layer has a trigger-condition that is a Boolean formula over the available sensory $exists$-predicates and a movement-action a. If a layer's formula evaluates to $True$, the layer produces output a; otherwise, it produces the special output null. Given a set of layers L, we will assume that the formula in each layer has length at most f and no two layers compute the same Boolean function and produce the same output.
- Relative to the total order on the layers, a layer i can have subsumption links to any layer j that is lower than i in the ordering; between any two layers, there can exist an output-inhibition or output-override link (but not both). An output-inhibition link from a layer L to a layer L' makes the output of L' null if the output of L is non-null. The set of output-override links to a layer L' are assumed to be in a total order, and the output of L' is either the value of the highest non-null layer-override link in the total order, if there is an output override link whose value is non-null, and the output specified by L' otherwise. The output of any layer that subsumes at least one lower-level layer is not available directly for output; otherwise, that layer's output is available.

The output of a set of ordered layers with subsumption links will be that of the highest layer relative to the order that is both available and non-null.

Relative to such an architecture, we will consider the following types of architecture modifications for reconfiguration:

1. Adding or deleting at most c_s subsumption links; and
2. Moving, deleting, or adding at most c_l layers, where added layers come from a layer-library M.

These modifications correspond to those considered in [2, 6, 17]. Note that any layer-move deletes all subsumption links between the moved layer and layers that are now above it in the layer-ordering and that any layer deletion removes all links to and from that layer.

2.2 Formalizing Reactive Swarms

A reactive swarm S is a group of reactive robots as specified in Section 2.1. Given a world W and a reactive swarm S, each freespace in W can hold at most one member of S and individual robots can only occupy freespace squares. A particular controller can be associated with more than one robot in S. This will be useful in assessing the computational burden (if any) of allowing robots with different control-behaviours in swarms.

To fully describe how a reactive swarm acts in a world, three policies must be specified:

1. When can individual robots move, *e.g.*, do all robots move relative to a shared clock (**synchronized movement**) or can individual robots move at arbitrary times relative to each other (**asynchronous movement**)?;
2. In what manner do robots in the swarm recognize and communicate with each other?; and
3. If two robots attempt to occupy the same square, how is this resolved?

For simplicity, we assume synchronous movement, no individual robot recognition (*i.e.*, all robots look alike regardless of the direction in which they are viewed) and no inter-robot communication of any type (outside of viewed robot positions and actions), and no movement conflict resolution (*i.e.*, if at any point during a task two robots in a swarm attempt to occupy the same space, that task terminates in failure). Though these choices (especially the first and third) are not realistic, they do constitute a special case that underlies more realistic cases (see Section 5.2), and hence are a good starting point for analysis.

2.3 Formalizing Reactive Swarm Morphogenesis

Our goal is to determine the computational complexity of robot swarm design and reconfiguration relative to a joint navigation / morphogenesis task. To allow investigation of different types of tasks, we will consider a basic type of morphogenesis in which one positioning of robots is transformed into another positioning, possibly in different parts of the world (which requires navigation). Let an **area** be any region of W and a **position** be an area a and an assignment of the members of S to specific freespace squares within a. For simplicity, we shall assume that areas are defined as four-sided polygons and the size of an area is the number of map-squares enclosed by that polygon.

The above yields the following four problems:

GIVEN REACTIVE SWARM MORPHOGENESIS (GRSM)
Input: A world W, a reactive swarm S, and initial and positions p_I and p_F for S in W.
Question: Can S move from p_I to p_F?

SELECTED REACTIVE SWARM MORPHOGENESIS (SRSM)
Input: A world W, a library A of reactive robots, a swarm-size $|S|$, and initial area a_I and final position p_F in W.
Output: A reactive swarm S of size $|S|$ selected from A and a positioning p_I of S in a_I such that S can move from p_I to p_F, if such an S and p_I exist, and special symbol $\perp$ otherwise.

GIVEN REACTIVE SWARM MORPHOGENESIS
WITH RECONFIGURATION (GRSM-REC)
Input: A world W, a reactive swarm S, initial and final positions p_I and p_F for S in W, a library M of layers, and positive integers c_l and c_s.
Output: A reactive swarm S' derived from S by at most c_l movements, deletions, and/or additions of layers relative to M and c_s additions and/or deletions of subsumption links that is able to move from p_I to p_F in W, if such an S' exists, and special symbol $\perp$ otherwise.

SELECTED REACTIVE SWARM MORPHOGENESIS
WITH RECONFIGURATION (SRSM-REC)
Input: A world W, a swarm-size $|S|$, initial are a_I and final position p_F in W, a library M of layers, a library A of reactive robots, and positive integers c_l and c_s.
Output: A reactive swarm S of size $|S|$ selected from A with at most c_l subsequent movements, deletions, and/or additions of layers relative to M and c_s subsequent additions and/or deletions of subsumption links and a positioning p_I of S in a_I such that S is able to move from p_I to p_F in W, if such an S and p_I exists, and special symbol $\perp$ otherwise.

Note there are no optimality restrictions in any of these problems, *e.g.*, the paths travelled by the swarm need not be the shortest possible.

The most basic problem above, GRSM, nicely illustrates how computationally different reactive swarms are from individual reactive robots. When a swarm consists of a single robot, GRSM is solvable in polynomial time by simulating the action of the robot for at most a polynomial number of timesteps [22, Lemma 1, Supplementary Materials]. However, this straightforward approach cannot work in general.

Result GRSM.A: There are instances of GRSM that require an exponential number of timesteps to proceed from the specified p_I to p_F.

This is perhaps not surprising given known difficulties in predicting the long-term behavior of dynamical systems [3]. A detailed analysis of the computational complexity of GRSM will be given elsewhere. For now, let us focus on those instances of our remaining three problems for which the basic swarm movement problem modeled by GRSM can be solved in polynomial time. This property is encapsulated in the following definition.

Definition 1. Given world W, a reactive swarm S, and initial position p_I and final area p_F in W, S is *polynomial-time verifiable* if S either does or does not proceed from p_I to p_F within a number of time steps that is polynomial in the size of W and S.

In the remainder of this paper, we will restrict SRSM, GRSM-REC, and SRSM-REC such that the reactive swarms created by these problems are polynomial-time verifiable.

3. REACTIVE SWARM MORPHOGENESIS IS INTRACTABLE

Following general practice in Computer Science [15], we define tractability as being solvable in the worst case in time polynomially bounded in the input size. We show that a problem is not polynomial-time solvable, *i.e.*, not in the class P of polynomial-time solvable problems, by proving it to be at least as difficult as the hardest problems in problem-class NP (see [15] for details).

Result SRSM.A: SRSM is NP-hard.

Result GRSM-REC.A: GRSM-REC is NP-hard.

Result SRSM-REC.A: SRSM-REC is NP-hard.

Modulo the conjecture $P \neq NP$ which is widely believed to be true [13], the above shows that the simplest versions of reactive swarm morphogenesis considered here are not solvable in polynomial time for all inputs. Note that this intractability holds for GRSM-REC and SRSM-REC even when the swarm consists of a single robot. This is perhaps unsurprising given the results in [22]. However, it does set the stage for our subsequent investigation into what restrictions can and cannot render these three problems tractable.

4. A METHOD FOR IDENTIFYING TRACTABILITY CONDITIONS

A computational problem that is intractable in general may yet be tractable relative to restrictions on the input. This insight is based on the observation that some NP-hard problems can be solved by algorithms whose running time is polynomial in the overall input size and non-polynomial only in some aspects of the input called **parameters**. The following definition states this idea more formally.

Definition 2. Let Π be a problem with parameters $k_1, k_2, \ldots$. Then Π is said to be **fixed-parameter (fp-) tractable**

Table 1: Parameters for Reactive Swarm Morphogenesis Problems

Parameter	Description
$\|L\|$	Max # layers in robot
E	# square-types in environment
f	Max length of layer-activation formula
r	Perceptual radius of robot
$\|S\|$	# robots in swarm
h	Max # controller-types in swarm
$\|A\|$	# robots in library A
$\|M\|$	# layers in library M
c_l	Max # layer-changes in robot
c_s	Max # sub-link-changes in robot
$\|a_I\|$	Size of initial area
$\|a_F\|$	Size of final area

for parameter-set $K = \{k_1, k_2, \ldots\}$ if there exists at least one algorithm that solves Π for any input of size n in time $f(k_1, k_2, \ldots)n^c$, where $f(\cdot)$ is an arbitrary function and c is a constant. If no such algorithm exists then Π is said to be **fixed-parameter (fp-) intractable** for parameter-set K.

In other words, a problem Π is fp-tractable for a parameter-set K if all superpolynomial-time complexity inherent in solving Π can be confined to the parameters in K.

There are many techniques for designing fp-tractable algorithms [7, 10], and fp-intractability is established in a manner analogous to classical polynomial-time intractability by proving a parameterized problem is at least as difficult as the hardest problems in one of the problem-classes in the W-hierarchy $\{W[1], W[2], \ldots\}$ (see [10] for details). Additional results are typically implied by any given result courtesy of the following lemmas:

Lemma 1. *[21, Lemma 2.1.30] If problem Π is fp-tractable relative to parameter-set K then Π is fp-tractable for any parameter-set K' such that $K \subset K'$.*

Lemma 2. *[21, Lemma 2.1.31] If problem Π is fp-intractable relative to parameter-set K then Π is fp-intractable for any parameter-set K' such that $K' \subset K$.*

Observe that it follows from the definition of fp-tractability that if an intractable problem Π is fp-tractable for parameter-set K, then Π can be efficiently solved even for large inputs, provided only that the values of all parameters in K are relatively small. This strategy has been successfully applied to a wide variety of intractable problems (see [10, 19] and references). In the next section we investigate how the same strategy may be used to render our various versions of reactive swarm morphogenesis tractable.

5. WHAT MAKES REACTIVE SWARM MORPHOGENESIS TRACTABLE?

Our reactive swarm morphogenesis problems have a number of parameters whose restriction could render these problems tractable in the sense defined in Section 4. An overview of the parameters that we considered in our fp-tractability analysis is given in Table 1. These parameters can be divided into four groups:

1. Restrictions on robot structure ($|L|$, $|E|$, f, r);
2. Restrictions on swarm structure ($|S|$, h);
3. Restrictions on robot / swarm reconfigurability ($|A|$, $|M|$, c_l, c_s); and
4. Restrictions on initial and final morphogenetic states ($|a_I|$, $|a_F|$).

We will assess the fixed- parameter tractability of reactive swarm morphogenesis relative to all parameters in Table 2 (Section 5.1), show how these results apply in more general settings (Section 5.2), and discuss the implications of these results for swarm engineering (Section 5.3).

5.1 Results

Our parameterized intractability results are summarized in Table 2. Each row describes an fp-intractability result that holds relative to the set of all parameters whose entries in that row are not dashes ("–") or X's (which mean that the parameters in those entries are of unbounded value or not applicable, respectively); if the result holds when a non-dashed parameter has constant value c, this indicated by an entry for that parameter with the value c. Observe that for each problem, fp-intractability holds not only relative to each of the parameters but to many combinations of these parameters, even when many of them are restricted to constant values.

At present, we have the following tractability results:

Result SRSM.E: SRSM is fp-tractable for $\{|A|, |a_I|\}$, $\{|A|, |a_F|\}$, $\{|E|, f, |a_I|\}$, $\{|E|, f, |a_F|\}$, $\{|E|, r, |a_I|\}$, and $\{|E|, r, |a_F|\}$.

Result GRSM-REC.I: GRSM-REC is fp-tractable for $\{|L|, |S|, |M|\}$, $\{|E|, f, |S|\}$, and $\{|E|, r, |S|\}$.

Result SRSM-REC.I: SRSM-REC is fp-tractable for $\{|L|, |A|, |M|, |a_I|\}$, $\{|L|, |A|, |M|, |a_F|\}$, $\{|E|, f,$ $|a_I|\}$, $\{|E|, f, |a_F|\}$. $\{|E|, r, |a_I|\}$, and $\{|E|, r, |a_F|\}$,

Note that the parameter-sets in all of these results are minimal in the sense that no parameter in any set can be deleted to yield fp-tractability.

5.2 Generality of Results

Our intractability results, though defined relative to simple robot and swarm architectures and tasks, have a much broader applicability. Observe that the architectures and tasks for which these results hold are in fact restricted versions of more realistic alternatives, *e.g.*,

Table 2: Fixed-parameter Intractability Results for Reactive Swarm Morphogenesis Problems

Result		Parameter											
		$\|L\|$	$\|E\|$	f	r	$\|S\|$	h	$\|A\|$	$\|M\|$	c_l	c_s	$\|a_I\|$	$\|a_F\|$
SRSM	B	4	5	–	–	p	1	1	X	X	X	–	p
	C	3	–	13	2	p	p	–	X	X	X	p	p
	D	3	5	–	–	p	p	–	X	X	X	p	p
GRSM-REC	B	3	3	–	–	–	2	X	1	p	0	–	–
	C	4	–	13	2	p	p	X	–	p	0	p	p
	D	4	5	–	–	p	p	X	–	p	0	p	p
	E	p	–	1	0	1	1	X	–	p	0	1	1
	F	p	5	–	–	1	1	X	–	p	0	1	1
	G	–	–	3	1	1	1	X	0	p	0	1	1
	H	–	5	–	–	1	1	X	0	p	0	1	1
SRSM-REC	B	4	5	–	–	p	1	1	0	0	0	–	p
	C	3	–	13	2	p	p	–	0	0	0	p	p
	D	3	5	–	–	p	p	–	0	0	0	p	p
	E	p	–	1	0	1	1	1	–	p	0	1	1
	F	p	5	–	–	1	1	1	–	p	0	1	1
	G	–	–	3	1	1	1	1	0	p	0	1	1
	H	–	5	–	–	1	1	1	0	p	0	1	1

- the deterministic reactive robot architecture is a special case of more complex architectures that have memory, probabilistic operation, and/or the ability to manipulate the environment, *e.g.*, move objects;
- the swarm architecture in which operation is synchronous, robots cannot communicate with or identify each other, and robot collisions are not handled is a special case of more complex architectures which are asynchronous, have identifiable and/or communicating robots, and/or handle robot collisions; and
- the joint navigation / morphogenesis task is a special case of more complex tasks involving swarm navigation and/or structure formation.

Intractability results for these alternatives then follow from the observation that intractability results for a problem Π also hold for any problem Π' that has Π as a special case (suppose Π is intractable; if Π' is tractable by algorithm A, then A can be used to solve Π efficiently, which contradicts the intractability of Π — hence, Π' must also be intractable).

Our fp-tractability results are more fragile, as innocuous changes to worlds, tasks, or architectures may in fact violate assumptions critical to the operation of the algorithms underlying these results. Hence, they may only apply to certain more complex cases, and this needs to be assessed on a case-by-case basis.

5.3 Discussion

We have found that designing and reconfiguring reactive swarms to perform a joint navigation / morphogenesis task in a known world is NP-hard (Results SRSM.A, GRSM-REC.A, and SRSM-REC.A). Our results immediately imply that it is unlikely that deterministic polynomial-time methods exist for any of these problems. There are also other useful implications. It is widely believed that $P = BPP$ [23, Section 5.2] where BPP is considered the most inclusive class of problems that can be efficiently solved using probabilistic methods (in particular, methods whose probability of correctness can be efficiently boosted to be arbitrarily close to probability one). Hence, our results also imply that unless $P = NP$, there are no probabilistic polynomial-time methods which correctly design or reconfigure robot swarms with high probability for all inputs. Together with the above, this suggests that *no* currently-used method (including the probabilistic methods employed in evolutionary robotics [18]) can guarantee both efficient and correct operation for all inputs for these problems.

As described in Section 4, efficient correctness-guaranteed methods may yet exist relative to plausible restrictions on the input and output. This is the strategy advocated by Francesca et al [14]. It seems reasonable to conjecture that some restrictions relative to the parameters listed in Table 2 should render our problems tractable. However, no single one or indeed many possible combinations of these restrictions can yield tractability, even when many of the parameters involved are restricted to very small constants (Results SRSM.B–D, GRSM-REC.B-H, SRSM-REC.B-H). Of particular interest here are the intractability of all of our problems when individual robot controllers are very simple (Results SRSM.C, GRSM-REC.C, and SRSM-REC.C), identical (Results SRSM.B, GRSM-REC.E, and SRSM-REC.B), or both (Results SRSM.B and SRSM-REC.B) and of our reconfiguration problems when there is only a single robot in the swarm (Results GRSM-REC.E–H and SRSM-REC.E-H). The latter highlights the computational power (and attendant design difficulty) of allowing robot controller modification. However, as shown by Results SRSM.B and SRSM-REC.B, there a comparable power and difficulty associated with the initial positioning of the robots in a swarm. As both of these sets of results depend on a suitably structured (and arguably complex) world, it would be interesting to know to what extent intractability can be preserved by trading a simpler world against a more complex swarm architecture incorporating robot identifiability and/or communication.

To date, though none of the restrictions considered here by itself yields tractability for any or all of our three problems, we have found a number of restriction-combinations that do (Results SRSM.E, GRSM-REC.I, and SRSM-REC.I). This is not indicative of a corresponding number of fundamentally different approaches to tractability – rather, each such combination effectively limits the number of possible swarm compositions and placements in the world that need to be considered, either explicitly ($|A|$ for composition, $|S|$, $|a_I|$, and $|a_F|$ for placement) or implicitly ($|L|$, $|M|$, $|E|$, f, and r for composition). The latter are of particular interest (especially $|E|$, f, and r), in that they suggest that one can get tractability in design by limiting the complexity of either the actual world (by restricting the structure in the world that is available to be perceived and exploited by the swarm) or the perceived world (by "dumbing down" the sensors ($|E|$, r) or the combinations of sensed information that can be acted on (f)). As such, these results are a first step towards investigating the computational tradeoff between world and swarm complexity mentioned above.

Three valid objections to our tractability results are that (1) the algorithms underlying these results are crude and rely on brute-force search, (2) the running times of these underlying algorithms are (to be blunt, ludicrously) impractical, and (3) the parameters involved may not be of small value in inputs encountered in practice. Objections (1) and (2) are often true of the initial fp-algorithms derived relative to a parameter-set. However, once fp-tractability is established, surprisingly effective parameterized algorithms are often subsequently developed with both greatly diminished non-polynomial terms and polynomial terms that are quadratic and even linear in the input size (see [7, 10] and references). Though Objection (3) initially appears to be more problematic, it is an artifact of our analysis process. We focused on minimal fp-tractable parameter-sets so that they would imply the maximum possible number of other fp-tractability results by Lemma 1. This minimization was accomplished by (1) replacing parameters in the running time with loose upper bounds stated in terms of other parameters, *e.g.*, $x \rightarrow f(y)$ where $x << f(y)$, and (2) replacing polynomial-terms formed of two parameters with an exponential term in a third parameter, *e.g.*, $x^y \rightarrow 2^z$. This obscures parameters whose values are small (particular if the parameters are the polynomial powers in (2)). Hence, the impractical algorithms currently underlying our fp-tractability results should be seen as promissory notes on practical algorithms involving larger sets of parameters that will be developed in future.

A final very important proviso is in order – namely, as illuminating as the results given here are in demonstrating basic forms of (in)tractability for swarm design and reconfiguration problems, these results do not necessarily imply that methods currently being applied to design swarms are impractical. Differing robot and swarm architectures, the particular situations in which these methods are being applied, and accepted standards by which method practicality is assessed may render the results given here irrelevant. For example, current methods may already be implicitly exploiting problem restrictions such that both efficient and correct operation (or operation that is correct with probability very close to one) are guaranteed. That being said, not knowing the precise conditions under which such practicality holds could have damaging consequences, *e.g.*, drastically slowed swarm creation time and/or unreliable swarm operation, if these conditions are violated. Given that reliable swarm operation is very important and efficient swarm design and reconfiguration is at the very least desirable, the acquisition of such knowledge via a combination of rigorous empirical and theoretical analyses should be a priority. With respect to theoretical analyses, it is our hope that the techniques and results in this paper comprise a useful first step.

6. CONCLUSIONS

In this paper, we have presented a formal characterization of reactive swarm design and configuration for a basic navigation / morphogenesis task in a known world. Our complexity results reveal that while these problems are computationally intractable in general, there are restrictions that render them tractable. Knowledge of these and other such restrictions should be useful in creating swarm design and reconfiguration methods that are efficient and correct.

There are several promising directions for future research. The first of these will be to continue our analysis relative to the currently examined robot and swarm architecture, as it provides a simple setting in which to address questions such as the computational tradeoff between world- and swarm-structure raised in Section 5.3 and the computational effects of allowing random initial robot positions (a first step in looking at more desirable forms of morphogenesis). The second will be to extend our analysis to incorporate more realistic robot and swarm models. Of particular interest here are robots using the probabilistic finite-state automaton formalism popular in the literature; at the very least, analysis of such a formalism would allow us to address the generality of the restrictions proposed in [14]. Last but certainly not least, derived fp-algorithms should be implemented and tested using actual robot swarms. Though difficult, such testing will be invaluable in providing guidance for structuring future theoretical analyses along the lines sketched here as well as future research in swarm engineering in general.

7. ACKNOWLEDGMENTS

I would like to thank Pim Haselager and Andrew Vardy for introducing me to the joys of reactive and swarm robotics, respectively, I would also like to thank the three reviewers for comments that helped in improving the presentation of this paper. This work was supported by NSERC Discovery Grants RGPIN 228104-2010 and 228104-2015.

8. REFERENCES

[1] L. Adleman, Q. Cheng, A. Goel, M.-D. Huang, D. Kempe, P. De Espanes, and P. Rothemund. Combinatorial optimization problems in self-assembly. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 23–32, 2002.

[2] R. Arkin. *Behavior-Based Robotics.* MIT Press, 1998.

[3] C. L. Barrett, H. B. Hunt, M. V. Marathe, S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Complexity of

reachability problems for finite discrete dynamical systems. *Journal of Computer and System Sciences*, 72(8):1317–1345, 2006.

[4] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[5] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[6] J. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2719–2724, Nice, France, 1992.

[7] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms.* Springer, 2015.

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

[9] R. Doursat, H. Sayama, and O. Michel. A review of morphogenetic engineering. *Natural Computing*, 12(4):517–535, 2013.

[10] R. Downey and M. Fellows. *Fundamentals of Parameterized Complexity.* Springer, Berlin, 2013.

[11] P. Dunne, M. Laurence, and M. Wooldridge. Complexity results for agent design. *Annals of Mathematics, Computing & Teleinformatics*, 1(1):19–36, 2003.

[12] H. Fernau, T. Hagerup, N. Nishimura, P. Ragde, and K. Reinhardt. On the parameterized complexity of the generalized Rush Hour puzzle. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 6–9, 2003.

[13] L. Fortnow. The Status of the P Versus NP Problem. *Communications of the ACM*, 52(9):78–86, 2009.

[14] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni, and M. Birattari. AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112, 2014.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W.H. Freeman, 1979.

[16] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. 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):76–88, 1984.

[17] D. Lyons and A. Hendricks. Planning as incremental adaptation of a reactive system. *Robotics and Autonomous Systems*, 14(4):255–288, 1995.

[18] S. Nolfi and D. Floreano. *Evolutionary Robotics.* MIT Press, 2000.

[19] U. Stege. The Impact of Parameterized Complexity to Interdisciplinary Problem Solving. In H. Bodlaender, R. Downey, F. Fomin, and D. Marx, editors, *The Multivariate Algorithmic Revolution and Beyond*, number 7370 in Lecture Notes in Computer Science, pages 56–68. Springer, Berlin, 2012.

[20] I. A. Stewart. The complexity of achievement and maintenance problems in agent-based systems. *Artificial Intelligence*, 2(146):175–191, 2003.

[21] T. Wareham. *Systematic Parameterized Complexity Analysis in Computational Phonology.* Ph.D. thesis, University of Victoria, 1999.

[22] T. Wareham, J. Kwisthout, P. Haselager, and I. van Rooij. Ignorance is bliss: A complexity perspective on adapting reactive architectures. In *Proceedings of the First Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics*, volume 2, pages 1–5, 2011.

[23] A. Wigderson. P, NP and mathematics — A computational complexity perspective. In *Proceedings of ICM 2006: Volume I*, pages 665–712, Zurich, 2007. EMS Publishing House.

[24] M. Wooldridge and P. E. Dunne. The computational complexity of agent verification. In *Intelligent Agents VIII*, pages 115–127. Springer, 2002.

APPENDIX

A. SKETCHES OF RESULT PROOFS

The proof of Result GRSM.A uses the simulation of the numerical and carry bits of an n-bit binary counter by a $(2n+1)$-member robot swarm appropriately placed in an $(n+4) \times 5$ world. Positions p_I and p_F correspond to the encodings of the numbers 0 and $2^n - 1$ in the counter, and it takes the robot swarm more than $2^n - 1$ timesteps to proceed from p_I to p_F.

All of our intractability results are derived using reductions. A reduction from a problem X to a problem Y is essentially a pair of polynomial-time algorithms A_1 and A_2 such that A_1 transforms an instance I of X into an instance I' of Y and A_2 transforms any answer to I' into an answer for I. Given a reduction from X to Y, if Y is solvable in polynomial time by an algorithm A', then A_1, A' and A_2 can be chained together to give a polynomial time algorithm for X. Conversely, if X reduces to Y and X is not solvable in polynomial time then neither is Y (otherwise, a polynomial-time algorithm for Y could be used with the algorithms A_1 and A_2 in the reduction to solve X in polynomial time, which would contradict the intractability of X). Fixed-parameter intractability can be proved by an analogous reduction between parameterized problems $\{K\}$-X and $\{K'\}$-Y that in addition requires A_1 to create I' such that for each $k' \in K'$ in I', $k' = f(K)$ for some function f. By reducing from an NP-hard ($W[x]$-hard) problem, we can show (fp-) intractability provided that $P \neq NP$ ($FPT \neq W[x]$). Though neither of these class-inequality hypotheses has been formally proved, they are commonly accepted as true within the Computer Science community (see [10, 13, 15] for details).

All of our intractability results are proved by reductions from the following NP-hard problem:

DOMINATING SET [15, Problem GT2]
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Does G contain a dominating set of size $\leq k$, *i.e.*, is there a subset $V' \subseteq V$, $|V'| \geq k$, such that for all $v \in V'$, either $v \in V'$ or there is a $v' \in V'$ such that $(v, v') \in E$?

The parameterized problem $\{k\}$-DOMINATING SET is known to be $W[2]$-hard. 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.*, $v \cup \{u \mid u \in V \text{ and } (\mathrm{u}, \mathrm{v}) \in \mathrm{E}\}$. We assume below an arbitrary ordering on the vertices of V such that $V = \{v_1, v_2, \ldots, v_{|V|}\}$.

Though we have a number of intractability results, the reductions underlying these results were created by combining a small set of techniques. Sketches of these techniques in the order in which they appear in the result-series is as follows:

- **Result SRSM.B**: Construct a $|V| \times (|V| + 2)$ world in which column i, $1 \leq i \leq |V|$, corresponds to vertex v_i in V and row i, $2 \leq i \leq |V|$ encodes the complete neighbourhood of v_{i-1} by placing 0's and 1's in the appropriate columns. The first row is all 0's and the top row is all G's. The only controller-type will go up if it sees a 1-square directly above or any other robot in the row above and there is no robot in the row immediately below, and go right if it is on top of a G-square. A swarm of k of these robots is placed initially in the first row, and will only progress to the last row (and then accumulate on the right-hand side of that row) if columns in which the robots were initially placed in the first row correspond to a dominating set of size k in the given instance of DOMINATING SET.
- **Result SRSM.C**: Construct a $|V| \times (|V|^2 + |V|(|V| - 1))$ world in which each column corresponds to a vertex in V. In this reduction, the complete neighborhoods are encoded in the columns in a staggered fashion such that the first neighbourhood is encoded in rows 1 to $|V|$ of column 1, the second in rows $2|V| + 1$ to $3|V|$ of column 2, the third in rows $4|V|+1$ to $5|V|$ of column 3, and so forth. This $|V|$-height column-wise encoding of a complete neighbourhood has symbol v_i in square i if v_i is in the complete neighbourhood being encoded and symbol 0 otherwise. There are $|V|$ controller-types in library A, each corresponding to a distinct vertex in V. The controller associated with vertex v will (1) go up unless there is an empty square immediately below and another robot immediately below that, *i.e.*, it is leaving a robot in its column too far behind, and (2) go right if it is either on top of a v-square or there is already a robot in the column immediately to the right. A swarm of size k is selected from A and placed in the lowest k squares of column 1. This swarm will only reach the topmost k squares of column $|V|$ if the set of vertices associated with the selected robots is a dominating set of size k in the given instance of DOMINATING SET.
- **Result SRSM.D**: Modify the construction in Result SRSM.C such that each vertex-square in a column-wise encoding of a complete neighbourhood is now encoded in binary as a horizontal strip of $\lceil \log_2 |V| \rceil$ 0- and 1-squares to the left of that vertex-square.
- **Result GRSM-REC.B**: Modify the construction in Result SRSM.B such that the robot controller omits the layer l that allows it to go up if there is a 1-square immediately above as well as the layer that makes it go right if it is on a G-square, makes l the only layer in library M, and place $|V|$ robots in the first row of the world. The swarm will only reach the top row if layer l can be added to k robots which correspond to a set of vertices that are a dominating set of size k in the given instance of DOMINATING SET.
- **Results GRSM-REC.C and GRSM-REC.D**: Modify the constructions in Results SRSM.C and SRSM.D such that (1) library A contains a single type controller that can only go up as previously specified and go right only if there is already a robot in the column immediately to the right and (2) library M contains all of the $|V|$ layers that allow a robot to go to the right if it is on top of a particular v-square (in the case of SRSM.C) or to the immediate right of the binary encoding of the v-square (in the case of SRSM.D). The swarm can only reach p_F if k layers from M can are added to the swarm (not necessarily to k distinct robots) which correspond to a set of vertices that form a dominating set of size k in the given instance of DOMINATING SET.
- **Results GRSM-REC.E and GRSM-REC.F**: Modify the constructions in Results GRSM-REC.C and GRSM-REC.D such that all layers are now added to a single robot.
- **Result GRSM-REC.G**: Modify the construction in Result GRSM-REC.F to have a single controller-type which has an additional $|V|$ layer-pairs where each pair consists of a v-right layer from M and a layer inhibiting that v-right layer. The effect of adding the appropriate k layers from M to the controller in Result GRSM-REC.F can now be accomplished by deleting the appropriate k inhibition-layers in the modified controller.
- **Result GRSM-REC.H**: Modify the construction in Result GRSM-REC.G using the v-square binary encoding scheme in Result SRSM.D.

Results SRSM-REC-B-H are derived using slight variants of the constructions sketched above for SRSM.B and GRSM-REC.C-H, respectively. Note that the NP-hardness in Results SRSM.A, GRSM-REC.A, and SRSM-REC.A follows from any of the SRSM, GRSM-REC, and SRSM-REC constructions described above.

The fixed-parameter algorithms underlying Results SRSM-E, GRSM-REC.I, and SRSM-REC.I are relatively simple, being based on the brute-force combinatorics of generating all possible swarms and/or positionings. However, the mathematical derivations used to phrase runtimes in terms of minimal parameter-sets are intricate and algorithm-specific, and thus, unlike the the constructions described above, not amenable to brief summary. Hence, descriptions of these algorithms will be deferred to the full paper.