Thesis:
The Computational Complexity of Controller-Environment Co-design using Library Selection for Distributed Construction

by

Mesam Timmar

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of *Computer Science*

Department of *Computer Science*

Memorial University of Newfoundland

October 2018

St. John's Newfoundland

Abstract

Creating specified structures through the coordinated efforts of teams of simple autonomous robots is a significant problem in distributed robotics. All previous effort, both empirical and theoretical, has focused on the problems of designing either controllers or environments which, in tandem with given environments or controllers, built the specified structures. In this paper, we give the results of the first computational and parameterized complexity analyses of the controller-environment co-design problem in the simple case where teams of finite-state robots are designed by selecting controllers from a given library. We show that this problem cannot be solved efficiently in general or under a number of restrictions, and give the first restrictions under which this problem is efficiently solvable.

We also consider two elaborations on this problem. First, we analyze the controller-environment co-design problem under a new architecture in which robots have a transient memory. Second, we give the first definitions of and derive computational complexity results for stigmergy-related parameters for the controller-environment co-design problem.

Acknowledgements

I would like to express my profound gratitude to Dr. Todd Wareham, my research supervisor, for his academic guidance, untiring support, research grant and his valuable and constructive feedback of this research work. His readiness to accommodate me whenever possible and to help me throughout my research work is much appreciated.

I would like to thank the Department of Computer Science and the School of Graduate Studies for granting me grants to support my research work and for giving me access to their printing facilities. I would also like to acknowledge the Memorial University of Newfoundland for providing me with this fantastic research opportunity.

My heartfelt appreciation goes out to Haseeb Tehsin, Minaam Fasihi, and Abis Abbas for proof-reading my research work and providing me with useful critiques.

Finally, I wish to thank my family, especially my father and my friends, especially Asma Javed, Syed Raza, Haris Tanvir and Shane Skinner for their constant support and encouragement throughout my research work. This study would not have been possible without their help.

— Mesam Timmar

Contents

Abstract **ii**

Acknowledgements **iii**

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 3

1.3 Organization of Thesis 5

2 Previous and Related Work **7**

2.1 Swarm Robotics 7

2.1.1 Difference of Swarm Robotics from Classical Robotics 10

2.2 Swarm Robotics in Construction 11

2.3 Classical and Parameterized Complexity Analysis 13

2.4 Classical and Parameterized Complexity Analysis with respect to Swarm Robotics in Construction 18

3 Controller-Environment Co-Design Through Library Selection: Results for the Basic Problem 20

3.1 Controller-Environment Model 21

3.1.1 Modifications to SI Model 26

3.1.2 Example of Construction of a Target Structure 27

3.2 Results 29

3.2.1 Classical Results from Previous Work 30

3.2.2 Parameterized Results From Previous Work 38

3.2.3 New Parameterized Results 40

3.3 Discussion 58

4 Controller-Environment Co-Design Through Library Selection: Elaborations 62

4.1 Other Controller Architectures 63

4.1.1 Motivation 63

4.1.2 Main Result 65

4.1.3 Reduction from SI to New Architecture 71

4.1.4 Discussion 73

4.2 Stigmergy 74

4.2.1 Motivation 75

4.2.2 Parameters of Interest 76

4.2.3 Results 78

4.2.4 Discussion 80

5 Conclusions and Future Work **81**

5.1 Conclusions 81

5.2 Future Work 82

Bibliography **84**

List of Tables

3.1 Parameters for the controller-environment co-design problem. 30

3.2 Summary of results for *CoDesignLS*. 31

3.3 Summary of fp-intractability results for the controller-environment co-design problem 61

4.1 Summary of fp-intractability results for the controller-environment co-design problem relative to the new architecture 74

4.2 Stigmergy-related Parameters for the controller-environment co-design problem. 78

List of Figures

2.1 A class diagram shows a tractable class T, enclosed in class B. 15

3.1 Example of a 2d grid environment 28

3.2 Example of the state diagram of a controller 28

3.3 Environment after the robot with controller given in Figure 3.2 runs over the environment given in Figure 3.1 29

3.4 State Diagram of Clique checker robot used in Lemma 1 36

3.5 Environment used in the reduction in the proof of Lemma 1 and Theorems 3, 4 and 5. 37

3.6 State Diagram of Clique checker robot used in Theorem 4 44

3.7 State diagram for checking uniqueness in Theorem 5 49

3.8 State diagram for edge check in Theorem 5 50

3.9 The environment used in Theorem 6 55

3.10 State diagram of the controller for the first clause robot used in Theorem 6. 55

3.11 State diagram of the controller for the clause robot i, $2 \leq i \leq |C| - 1$, used in Theorem 6 56

3.12 State diagram of the controller for the last clause robot used in Theorem 6. 56

4.1 State diagram for controller used in proof of Theorem 8. 70

4.2 Environment used in proof of Theorem 8. 71

Chapter 1

Introduction

This chapter gives a brief introduction and motivation for the research done in this thesis. We start with the motivation in Section 1.1. In Section 1.2, we highlight the research questions that are answered in this research. Finally, Section 1.3 describes the organization of this thesis.

1.1 Motivation

Swarm robotics, which is a subfield of multi-robotics, is inspired by natural swarms of social insects [25]. Collective behavior and the self-organization of social insects have attracted the attention of scientists for a long time. The way termites construct their mounds and the phenomenon of pheromone trails used by ants for finding food are two examples of such natural swarms [5]. These biological swarms exhibit astounding features. Even though an individual member of a swarm does not seem very capable, the results of their collective efforts are impressive. Termites produce one of the most amazing structures among social insects. Their mounds have features like natural air

conditioning and a royal chamber for the queen to lay her eggs. One might assume that a leader, i.e., the termite queen guides such construction. However, research shows that the construction process of mounds is not guided by any individual termite but is a result of coordinated effort via stigmergy [5], which is a way of communication by making changes in the environment.

Following this mechanism, researchers are working on building swarms of relatively simple robots and making them work in certain environments to achieve various construction-related tasks [40]. Such environments are often enhanced by features to guide the robots to complete their tasks, e.g. markers in the environment for construction material or for the position of the target structure. One of the features of natural swarms is that if some ant or termite dies, this will not halt the functioning of other individuals. The swarm will adjust itself, and the process of searching for food or mound construction will carry on. Swarm robotics tends to mimic this approach of having a large number of simple agents in a team to gain robustness, flexibility, and scalability in the system. Compared to a single but powerful robot, teams of simple robots will prove vital for elevated risk missions [8] e.g., deep space missions or working in catastrophic conditions where it is impossible for human beings to operate or too risky to send expensive robots or machinery. A large number of simple robots thus become an ideal candidate to work under such conditions because such teams of robots are scalable and robust. So even if some members of a team malfunction, this will not jeopardize the overall functionality of the team. Such malfunctioning members can be replaced efficiently, economically and easily in a short span of time.

The problem of creating a team of robots for a given environment and the problem of designing an environment for a given team of robots to achieve a construction-

related task are intractable in general and also intractable relative to restrictions on a number of sets of parameters of the problem [37]. Hence, there is a need to explore variants of these problems which might be tractable. One such variant is to select robots for the team from a given library of controllers instead of creating them from scratch and to design the environment simultaneously with the team (**co-design problem**). In this thesis, we have studied this co-design problem using both previously derived [35, 37] and new [32] results.

1.2 Contributions

In this research, we have done the first classical and parameterized complexity analyses of the controller-environment co-design problem. The controller-environment architectures that we used to study the co-design problem are similar to the one presented in [37]; we refer to this model as the Swarm Intelligence (SI) model (defined in detail in Section 3.1). The basic entities in the model of structure creation by robot teams are environments, target structures, individual robots, and robot teams (along with target structure and team positioning). In this research, we have derived the computational and parameterized complexity analysis results for the above-stated co-design problem to answer the following questions (see Section 2.4 for further motivation):

1. **Is co-design efficiently solvable in general?** By Theorem 1 in Section 3.2.1, it has been shown that co-design is not efficiently solvable in general.

2. **If co-design is intractable in general, can it be made efficiently solvable by applying restrictions to parameter sets?** Results in Section 3.2 shows that co-design problem remains intractable even after applying restrictions to multiple parameters.

3. **What results hold if we use a different architecture for robot controllers?** We have considered a new architecture (defined in Section 4.1) which is an extension of the finite state controller architecture of the SI model defined in [37] (restated in Section 3.1), in which the robots have a transient memory. We have shown intractability under the new architecture even relative to the parameter combinations for which we could not get intractability for the SI model – that is, by just one reduction in Theorem 8 in Section 4.1.2 we have shown that the co-design problem is intractable under the new architecture relative to all the controller and team related parameters related to this problem given in Table 3.1.

4. **What results do we get for the co-design problem if we restrict parameters related to stigmergy?** We have done the first computational complexity analysis investigating aspects of stigmergy. We have defined several stigmergy-related parameters in Section 4.2.2 and derived some basic results relative to these parameters. Results in Section 4.2.3 shows that restricting several combinations of these parameters still does not make co-design problem tractable.

Note that the results cited in points 1 and 2 above were previously given (frequently without proof due to conference page limits) in [32].

1.3 Organization of Thesis

This thesis is organized as follows:

- In **Chapter 2**, we have given an overview of swarm robotics and their use in construction. An introduction of swarm robotics, motivation, main characteristics and comparison with classical robotics is given in Section 2.1. Section 2.2 discusses the use of classical robots and the need for swarm robotics in the construction industry. In Section 2.3, we explain the basics of classical and parameterized complexity analysis and how they are used to answer the questions that are of significant interest in our research. Section 2.4 talks about some of the previous computational complexity work done on the construction-related problems in swarm robotics.

- In **Chapter 3**, we investigate controller-environment co-design through library selection problem. In Section 3.1, the controller-environment model (SI model) used to study the co-design problem is given. We formalize the co-design problem in Section 3.1. Results for the co-design problem under the SI model are given in Section 3.2 and are discussed in Section 3.3. The parameters for the co-design problem that we considered in our proofs are given in Table 3.1, and a summary of all results in this chapter is given in Table 3.2.

- In **Chapter 4**, we have elaborated on the results given in Chapter 3. In Section 4.1, a newly proposed architecture is analyzed for the controller-environment co-design through library selection problem. Intractability for the co-design problem under this new architecture is proven in Section 4.1.2 and the relation-

ship between the two architectures is described in Section 4.1.3. Section 4.2 discusses stigmergy in detail and for the very first time parameters quantifying various aspects of stigmergy in the problem under study are defined and analyzed.

- In **Chapter 5** we summarize our work and give several important directions for future work.

Chapter 2

Previous and Related Work

This chapter gives an overview of swarm robotics, its motivation and use in the construction industry. We also explain some of the important concepts related to complexity theory and talk about some previous related work in the construction industry. Section 2.1 talks about swarm robotics in general and Section 2.1.1 highlights the differences between swarm robotics and classical robotics. In Section 2.2, we discuss the use of swarm robotics in construction. Section 2.3 gives an introduction to classical and parameterized complexity analysis and how these are used to answer the questions that are of significant interest in our research. Finally, Section 2.4 talks about the previous computational complexity work done on construction-related problems in swarm robotics.

2.1 Swarm Robotics

Swarm robotics is an area of robotics inspired by biological swarms [25]. To understand swarm robotics, we will first have a look into natural swarms of insects.

Biological swarms are mainly seen in the social insects like ants, termites, bees, etc. and also seen in communal animals like schools of fish and flocks of birds [10].

To live together in a society, insects have to distribute tasks and have to plan and decide where to live, where to collect the food from and how to build their nest. This results in a need for communication among insects in a society. A single insect does not have enough capability to either accomplish a task on its own or make a decision and lead other insects of the swarm. For this purpose, insects make collective decisions [4]. For example, when ants have to find a food source, the explorer ants spread out and when they find a food source they leave pheromones behind on their way back to nest. Other ants follow the trail and extract the food source [4]. A similar phenomenon is found in honey bees. When a honey bee finds a food source, it comes back to the nest and makes certain types of movements called the honey bee dance to communicate the location of the food source. Other bees then locate the food source and collect food [30]. Similarly, these insects with their collective efforts find and build new dwellings [4].

From above we see the main characteristics of biological swarms [25]:

- **Indirect Communication:** The agents of a biological swarm communicate with each other via making changes in their environment, i.e., via stigmergy [5]. An example of this is laying pheromone trail for finding food by ants [4].

- **Decentralized Decision Making:** There is no central governing authority in biological swarms that make important decisions; instead the decision making process is a result of the collective behavior of the members of swarm [4].

- **Robustness:** The members of a swarm carry equal importance in the colony.

No particular member has supreme abilities or power to run the swarm such that the rest of the swarm is dependent on that individual member. This characteristic makes the swarm robust which means that even if some of the members of a swarm die then it will not affect the functionality of rest of the colony rather, some other members will replace the dead ones.

Swarm intelligence tends to mimic these characteristics. Swarm robotics is a field of multi robotics in which a large number of simple robots work together to achieve some specified task [25]. The team of robots in swarm robotics have the characteristics mentioned above [19, 25] which means the following:

- Robots in the team do not communicate directly; instead, they communicate with each other by making changes in the environment, i.e., via stigmergy.
- Any central system does not control robots in a team; instead, the robots operate autonomously in their local environment with their local information.
- Individual robots do not have information about the whole swarm neither does an individual robot has enough capability to achieve the final goal on its own. Instead, the task is achieved by the collective effort of the team as done in biological swarms. So, if any member of the team somehow malfunctions, then it will not halt the overall functionality of the team and the rest of the team will continue working. This requires that the team should consist of a large number of robots so that malfunctioning robots can be replaced easily.

2.1.1 Difference of Swarm Robotics from Classical Robotics

In this section, we highlight the differences of swarm robotics from classical robotics. Classical robots (which we will refer to as Good Old Fashioned Robots (GOFR)) are different from swarm robots in many ways [19];

- If we look into individual capabilities of GOFR then it is apparent that an individual GOFR can perform very complex tasks e.g. robots used for surgical procedures [20] or self-driving cars [21]. On the other hand, individual robots in a swarm are dumb as compared to GOFR. They are not capable enough to achieve a task on their own but working as a team allows them to solve complex problems which are not solvable otherwise by an individual [19].

- GOFR usually do not work in teams. Instead, an individual GOFR is designated for a specific task of which it has either complete knowledge and it works on its own, or it is controlled by another system or human being. On the other hand, an individual robot in a swarm of robots is not capable enough to achieve complex tasks alone. Robots in swarm robotics accomplish a task while working in a team consisting of either same type of robots (**homogeneous team**) or with different types of robots (**heterogeneous team**). The number of robots in a team is usually large, as a large number of simple robots is often able to complete tasks better than a single complex robot [19].

- GOFR can work either on their own, in which case information is fed into the system, or GOFRs are either fully or partially controlled by some central controlling system or by humans in which case they have access to global information. On the other hand a swarm of robots is not controlled by any central

system; instead, the robots operate autonomously with their locally perceived information.

- GOFR are usually designed to solve specialized tasks, so their internal data structures and controllers are complex and are related to the task they are assigned. This makes them expensive and not easily replaceable in certain cases if they malfunction, e.g. working in deep space or in a mine where human access is not possible. Individual robots in swarm robotics do not have information about the whole swarm nor does an individual robot have enough capability to achieve the final goal on its own, which makes them cheap. The swarm is not dependent on any individual robot hence making the system robust and scalable, which is not easily achievable in GOFR based systems.

Swarm robotics has the advantages over GOFRs in terms of robustness, scalability, decentralization and self-organization [19]. In the next section, we will discuss how swarm robotics and GOFRs are being used in the construction industry.

2.2 Swarm Robotics in Construction

In recent decades we have seen an increase in the use of robots in many different fields of science and engineering [7]. The construction industry, in which previously there was not much research being done, has now been under the spotlight for researchers for the use of robots [2, 3, 23, 26]. There is very little practical use of GOFRs in the construction industry, and even when robots are used in construction, they are fully or partially supervised by human beings. The reason for the reluctance of the

construction industry for the use of GOFRs is that the construction environments are not structured, and therefore safe human-robot interaction is of major concern [1, 26]. GOFRs are designed to work in an organized environment where everything is in place for a GOFR to operate. Some examples of semi-autonomous robots in the field of structure creation are road pavers and asphalt compactors. Semi-autonomous robots are also used for interior finishing in house building [3].

In the construction industry, where even GOFRs have not made an enormous impact, swarm robotics is still merely an academic topic of research. Previous research includes the study of the unsupervised construction of structures by robots, most of which is inspired by natural swarms, e.g., wasp net construction [31]. By mimicking biological swarms, algorithms for the construction of specified structures using autonomous teams of robots have been designed [6] e.g robot teams that mimic the way termites build their mounds [40]. Most of these previously designed algorithms focus on the design of homogeneous robot swarms, i.e. all the robots in the teams have the same type of controller, with stochastic behavior rules which are necessary for robot teams to work in deadlock-free manner [6, 22, 27, 31, 39, 40]. However, this work does not deal with the controller environment co-design problem – that is, whether there exist any algorithm that can design a team of robots and also that team's operating environment at the same time for efficient completion of a target structure.

2.3 Classical and Parameterized Complexity Analysis

Two questions of major concern are raised in Section 1.2:

- Is co-design efficiently solvable in general?
- If co-design is intractable in general, can it be made efficiently solvable by applying restrictions to the parameter sets?

These are best answered using computational complexity theory, in particular through classical [15] and parameterized complexity analysis [12]. Theories of computational complexity are designed essentially to rule out whether certain kinds of algorithms do or do not exist. Here we give a brief introduction to some of the important concepts related to computational complexity theory [12, 15];

- **Tractable Class:** A class of problems that can be solved by algorithms whose running time is bounded by a certain kind of function of the size of the problem instance is called a tractable class. For example, a polynomial time tractable class (**class P**) has problems that can be solved by polynomial-time algorithms e.g. sorting a list, searching in an ordered or unordered list etc. The running time of algorithms that solve these polynomial-tractable problems is of the form $O(n^c)$ where n is the input size and c is some non-negative integer. Polynomial-time solvable problems are said to be solved efficiently.
- **Intractable Class:** A class containing problems for which efficient solvability is not possible i.e. there cannot exist any algorithm to solve such problems

whose running time is bounded by a certain kind of function of the size of the problem instance is known as intractable class. For example, some problems in a polynomial-time intractable class are not solvable in polynomial-time in terms of the size of the problem input. Note that an intractable class e.g. class B (from Figure 2.1) is either known to properly include the tractable class T or it is conjectured to properly include class T. For example, $EXPTIME$ (the class of problems that are solvable in exponential time) properly includes class P, whereas the class NP is conjectured to properly include class P i.e $P \neq NP$ [14].

- **Reduction:** A many to one reduction from problem A to problem B is essentially a transformation of an instance of A into an instance of B. The idea here is that if any input of problem A can be efficiently transformed into an input of problem B then any algorithm that solves B can be used in tandem with the reduction to solve A. The time that such a transformation algorithm takes is bounded such that some form of tractability is preserved.

- **Class Hardness and Completeness:** If for a class of problems A, there exists a problem C such that every problem in class A is reducible to C by an algorithm preserving A-time solvability then we call C an **A-hard** problem. If an A-hard problem is a member of class A then such a problem is called **A-complete**.

  In Figure 2.1, every problem in class B is reducible to each problem in class B-h; hence, B-h is essentially a class of B-hard problems. The class B-c refers to the class of B-complete problems i.e. B-hard problems that belong to class B

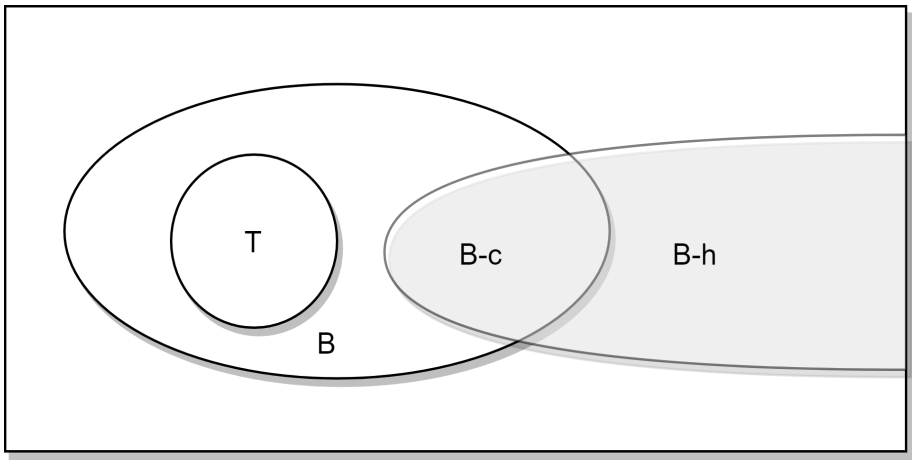

Figure 2.1: A class diagram shows a tractable class T, enclosed in class B. Here, B-h is the class of B-hard problems and B-c is the class of B-complete problems. Note that B-c is contained in B-h

as well. As an example, every problem in class NP is polynomial-time reducible to an NP-hard problem. If an NP-hard problem is a member of class NP then it is called an NP-complete problem.

In the case of polynomial time tractability, in Figure 2.1, T will be the class P, the intractable class B is NP, and B-h and B-c are the NP-hard and NP-complete classes respectively.

Now that we have described the basic concepts of tractable class, intractable class, reduction, and hardness and completeness, we can discuss what classical and parameterized complexity analysis are.

- **Classical Complexity analysis** [15]: Such an analysis focuses on polynomial-time tractable and intractable problems. It essentially rules out the possibility of the existence of polynomial-time algorithms for certain problems. It does this

by reducing a known polynomial intractable problem to a problem of interest for which intractability is to be proved. For example, NP-hard and NP-complete problems are reduced to a problem A to prove intractability of the problem A (subject to the conjecture $P \neq NP$). Note that in such an analysis, the reduction preserves polynomial time solvability i.e. the transformation function is polynomial time bounded.

- **Parameterized Complexity Analysis** [12]: A problem instance typically has multiple aspects or parts. We call each such aspect a **parameter**. For example, in Table 3.1, we have given several parameters of the controller-environment co-design through library selection problem. If a problem is polynomial-time intractable in general then there is still a possibility of solving such a problem efficiently by restricting the values of some parameters, i.e., efficient solvability is possible via **fixed-parameter tractable (FPT)** algorithm. We call a problem A **fixed-parameter tractable relative to a set of parameters** K (i.e $\langle K \rangle$-A **is fp-tractable**) if there exist an algorithm for A whose running time is upper bounded by time $f(K)n^c$, where n is the problem size, c is a constant, K is a parameter set of problem A, and f is some function of K. Hence, A can be efficiently solvable by such an FPT algorithm even for larger input size if the parameters in set K have small values.

  A problem for which an FPT algorithm relative to a set of parameters K is not possible is called a **fixed-parameter intractable** (**fp-intractable**) problem relative to parameter set K. The class XP properly encloses class FPT (the class that contains problems with FPT algorithms) and several classes in W-

hierarchy are conjectured to properly enclose class FPT e.g. $W[1]$ and $W[2]$ (see [12]). Problems that are XP-hard and $W[1]$ or $W[2]$ hard (relative to the conjecture that $FPT \neq W[1]$ and $FPT \neq W[2]$) are examples of fp-intractable problems.

Parameterized complexity analysis establishes whether fixed-parameter tractable (FPT) algorithms do or do not exist for polynomial-time intractable problems. It rules out the possibility of existence of FPT algorithms for a certain parameter set K of polynomial-time intractable problem A by a parameterized reduction from a known intractable class problem e.g. from a W-*hard* problem for some class W in W-hierarchy to an instance of A with constant values of parameters in K.

A **parameterized reduction** is a transformation which preserves fixed-parameter tractability i.e. the transformation function in the reduction runs in fixed-parameter tractable time and the parameter K in one problem is a function of the parameters in the other problem in the reduction. Such a reduction is used to prove hardness for problems relative to classes $W[1], W[2]$, and XP in the W-hierarchy [12].

In a parameterized complexity analysis, we often have a group of parameters for a particular problem, and we are interested not only in whether the problem is fp-tractable or fp-intractable relative to individual parameters but also relative to various combinations of the parameters. One way of displaying these combinations and their fp-tractability and fp-intractability results is to list the parameters involved in each result. Another way of displaying such combinations

is by an **Intractability Map** [33]. In an intractability map, the combinations of parameters correspond to the entries in a 2-dimensional table, and for each combination, we state whether it is fp-tractable, fp-intractable or unknown. Examples of such maps are given in Tables 3.3 and 4.1.

Going back to the two questions asked at the beginning of this section, from the above discussion we established that the first question is answered using classical complexity analysis and the second one through parameterized complexity analysis. In this thesis, we have first established that the controller-environment co-design through library selection problem is intractable in general i.e. polynomial-time intractable, in Theorem 1 in Section 3.2.1. In Sections 3.2.2, 3.2.3 and 4.1.2, we establish fp-intractability for this co-design problem relative to various sets of parameters given in Table 3.1. At the end of the Section 3.2.3 in Theorem 7, an fp-tractable algorithm is designed for the co-design problem relative to one particular set of parameters.

2.4 Classical and Parameterized Complexity Analysis with respect to Swarm Robotics in Construction

There is some complexity work previously done on the computational complexity of designing an autonomous multi-agent system that can perform specified tasks. Environments and control mechanisms are formalized generally and are powerful (e.g. Turing machines or Boolean propositional formulae) in the work done in [13, 28, 42]. Recent work includes more explicit models for robot controllers and the environments

in which the robots operate [34, 36, 37, 38]. However, none of these papers talk about the construction of target structures except Wareham and Vardy's paper [37] which has discussed controller design and environment design separately but did not discuss about designing them simultaneously (which is the co-design problem that we have studied in our thesis research) and Wareham's paper [35] which talks about designing of robot teams through selection of controllers from a library of controllers for construction, repair and maintenance of structures. Both of these works are done relative to a simple model of the controller and a 2-d grid-based environment for structure creation. In [37], the authors considered the problem of verifying if a given controller-environment pair can create a specified structure. They also considered if it is possible to design a controller relative to a given environment or to design an environment in order to make the given controller work to complete a construction-related task. Given the general intractability of these problems, the authors of [37] hoped to achieve tractability for these problems if both the controller and environment are designed simultaneously and the controllers are selected from a given library of controllers instead of creating from scratch. However, this proved not to be the case as shown by the work done in [32], which is the basis of the results in Chapter 3 of this thesis.

Chapter 3

Controller-Environment Co-Design Through Library Selection: Results for the Basic Problem

In this chapter, we have formalized the controller-environment co-design (through library selection) problem and have derived intractability results and a tractable algorithm for it. First of all, in Section 3.1, we describe the controller-environment model for our co-design problem in detail and discuss all the entities involved in it. Then in Section 3.2, we give intractability results and one tractability result for our co-design problem. In the end, we discuss what all these results presented in Section 3.2 mean in the discussion Section 3.3.

3.1 Controller-Environment Model

In this section, we first review the basic entities in the model of structure creation by robot teams given in [37]. We will refer to this model as the **SI (Swarm Intelligence) model** because [37] was published in the journal *Swarm Intelligence*. The basic entities in the SI model are environments, target structures, individual robots, and robot teams, which are described below. Later in this section, we formalize the computational problem of controller-environment design under library selection for construction that we will analyze in the remainder of this chapter.

- **Environments and Target Structures**: Our robots operate in a finite 2D square-based environment E in which each square has a square-type drawn from a set E_T. Examples of such environments can be seen in Figures 3.1, 3.3 and 3.5. 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 southern-most west-most (lower left) corner of E. A structure X in an environment E is a two-dimensional pattern of squares in an $m \times n$ grid whose location in E is specified relative to the position p_X of the lower left corner of the grid. Environment types in E_T mimic real world environment features like sand, grass or gravel etc. Figure 3.3 shows an environment in which a target structure is a combination of square blocks denoted by square-type X. The target structure in Figure 3.3 is a result of running a robot with the controller given in Figure 3.2 on the environment in Figure 3.1.

- **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). 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 $r \leq 1$ of the robot's current position to type e via predicates of the form $enmod(e, pos)$ where pos is specified as for $enval()$.

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 between states q and q' has a propositional logic trigger-formula f based on the predicates $enval()$ described above (see Figure 3.2). If a transition's trigger-formula evaluates to $True$, this causes a symbol x to be written by the predicate $enmod()$ described above and the robot's state to change from q to q'. Transition with $f = *$ executes if no other non-$*$ transition executes; if $x = *$, no symbol is written.

As multiple transitions could be enabled at the same time which gives the option as to how a robot should operate in such a situation i.e probabilistically, non-deterministically etc. For simplicity, the authors in [37] restricted robot

operations so that only one transition could be triggered at any given time relative to robot's current state and the environment type that the robot sensed, which makes the robot operation deterministic. Later in this section, we will give the definition of determinism used in [37].

- **Robot teams**: A team T consists of a set of the robots described above, where there may be more than one robot with the same controller in a team. 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 team attempt to occupy 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 subset of $|T|$ squares in E. For simplicity, team members do not communicate with each other directly (though they may communicate indirectly through changes they make to the environment, i.e., via stigmergy [5]). Team members can move either synchronously or asynchronously as specified; however, in both cases, once movement is triggered, it is instantaneous and atomic in the sense that the specified movement is completed.

The time that a team of robots takes to complete any given construction task is crucial in the real world. We restrict our robot teams to the construction tasks that are completed quickly instead of making robot teams operate over unlimited periods of time which results in intractability [37, Section 3]. Also, for reliable construction of a structure by a team of robots, we force the team operations to be deterministic. Following are the definitions of determinism and time-bounded completability as given in [37].

- **Determinism**: Determinism has been defined differently by different authors. The traditional definition of determinism for finite-state automata is that an automaton can sense a single symbol at a time and each symbol maps to at most one action, which can be either change of state or triggering of an associated action [18, Section 2.2]. The authors in [37] defined determinism a bit differently for robot operations. In case of **synchronous** team operations, a robot in a team is allowed to perform an action at the common clock ticks and it is allowed to perform an action at arbitrary times if team operations are **asynchronous**. The following rules apply when an FSR in the SI model is allowed to perform an action relative to robot's current state q:

  1. If a transition $t = \langle q, f, c, m, q' \rangle$ is enabled i.e f is satisfied, and no other transition is enabled then t is executed. This means that the robot whose current state is q, will make changes c to the environment, if $c \neq *$ and perform movement action m (which can be either goNorth, goSouth, goEast, goWest or stay) and will change its state to q'.

  2. If no transition is enabled then the default transition with $f = *$ is executed, if such a transition with $f = *$ is defined for q.

  3. If at any time, more than one transition is enabled i.e. transition-triggering formula f for multiple transitions is satisfied then the execution of task being performed by the robot and its team is terminated [1].

Finite-state robots (FSRs) can sense and be enabled by arbitrary patterns of

[1]Note that for the purposes of this thesis, these team operation rules and notion of determinism are broadened in Section 3.1.1.

squares within radius r of their current position. The number of such patterns that can be encountered is large and also variable as the sensed environment can change as the robot and other robots in the team can either move or make changes to the environment. Therefore, determinism is not defined relative to the actions of an individual robot; instead, the operation of an FSR can only be deterministic in terms of a particular FSR team working in a particular environment.

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

Now, as we have defined all the entities in SI model, we can formalize the computational problem of controller-environment co-design under library selection for construction that we have analyzed in our research as follows:

Controller-Environment Co-design under Library Selection

Input: A 2-d World-grid W, square-type set E_T, FSR library L, team-size $|T|$, structure X, initial positioning p_I of size $|T|$ in W, and position p_X of X in W.

Output: A team T of size $|T|$ selected from L and an environment E consisting of assignments of types from E_T to the squares of W such that T started at p_I in E creates X at p_X.

We will use $CoDesignLS$ as an acronym for this problem in the remainder of this doc-

ument. $CoDesignLS^{fast}$ with superscript $fast$ indicates that we will only be studying the instance of the problem in which robot team complete the construction task efficiently, which means the team's construction task is (c_1, c_2)-completable relative to some constants c_1 and c_2 as defined earlier. We will append subscript syn and asy with $CoDesignLS^{fast}$ to denote the problem instance with synchronous ($CoDesignLS^{fast}_{syn}$) and asynchronous ($CoDesignLS^{fast}_{asy}$) team operations respectively.

3.1.1 Modifications to SI Model

In our research, we have extended the SI model used in [37]. Notions of the deterministic and time-bounded robot and team operation were introduced in [37] (given in the previous section) to ensure that requested structures are created by robot teams reliably and quickly. In our research, we have broadened these notions as follows:

- **Determinism**: Contrary to the robot operation rules under the SI model given in previous section, where enabling of multiple transitions at any time resulted in termination of the execution of task ($3rd$ rule of robot operation), here we allow multiple transitions to be enabled at the same time if each transition writes the same symbol to the same position and changes the robot's state to the same state q' while making the same movement action; otherwise robot and team operation is terminated.

- (c_1, c_2)**-completability**: Instead of requiring that each robot team complete its task within $c_1|E|^{c_2}$ time-steps as in [37], we allow robot teams to complete their tasks within $c_1(|E| + |Q| + |f| + d)^{c_2}$ time-steps, where $|E|$, $|Q|$, $|f|$, and d are the number of squares in the environment, the maximum number of states, the

maximum trigger-formula length in any transition, and the maximum number of transitions per state in any controller in T.

An example of a construction using this modified version of SI model is discussed in the next section.

3.1.2 Example of Construction of a Target Structure

Before we dive into results for tractability and intractability, let's just run through a simple construction example using the SI model. An example construction task is described in Figures 3.1, 3.2 and 3.3. An initial environment has been shown in Figure 3.1 with $E_T = \{A, B, C, X\}$. Team T has a single robot, which has a controller selected from library L, where L has a single controller whose state-diagram is given in Figure 3.2. The position of the robot is shown by the symbol R in the environment. The robot is initially placed at $E_{1,2}$, i.e. the $1st$ column and $2nd$ row in E ($E_{1,2}$ has environment type C). Its initial position in the environment is denoted by $p_I = E_{1,2}$. The robot will move east from its initial position and will sense the environment type of the square-block underneath it. If the square-block type is A, the robot will change it to X. It will stop the construction process when it reads square-block type C, hence completing the construction task.

C	B	B	B	B	C
R	A	A	B	A	C
C	B	B	B	B	C

Figure 3.1: Example of a 2d grid environment with $E_T = \{A, B, C, X\}$. Here the position of robot is denoted by R in the left most column.

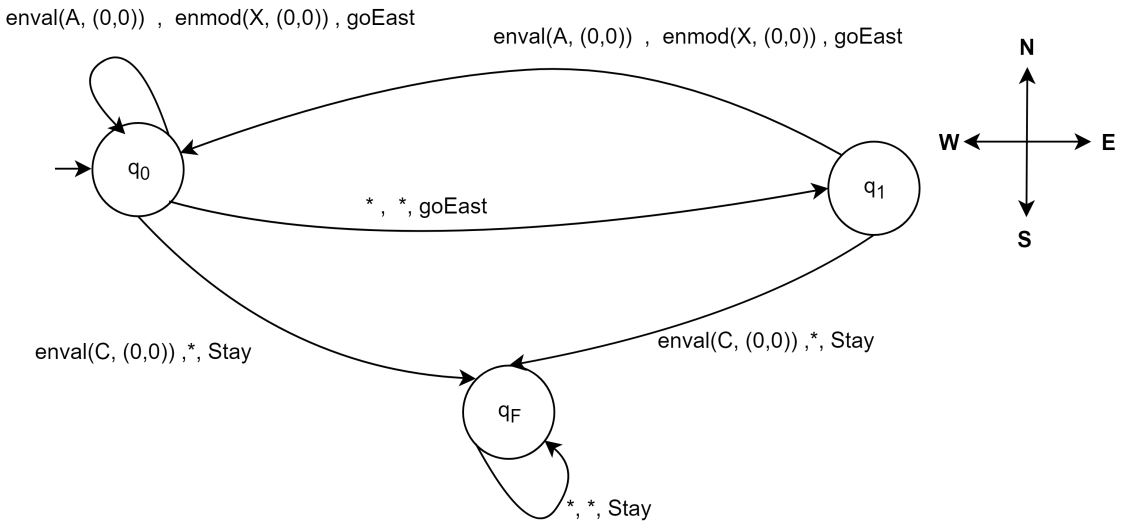

Figure 3.2: State Diagram of a controller that reads environment types in the given environment and keeps moving to east until it reads square-type C. If it reads square-type A, it replaces that with square-type X, hence creating the target structure.

C	B	B	B	B	C
C	X	X	B	X	R
C	B	B	B	B	C

Figure 3.3: Environment after the robot with controller given in Figure 3.2 runs over the environment given in Figure 3.1. Symbols of type X denote the resulting target structure created by robot. Here the final position of robot is denoted by R in the right-most column.

3.2 Results

In this section, we first discuss classical complexity results for *CoDesignLS*. If a problem is intractable in general then we need to consider what restrictions might make that problem tractable. Such restrictions are phrased in terms of aspects of our problem input or output; each such aspect is known as a parameter as defined in Section 2.3. The parameters analyzed for *CoDesignLS* in this paper are shown in Table 3.1 and can be broken into two groups:

1. Restrictions on robot teams and individual robots ($|L|, |T|, |Q|, d, |f|, r$) and
2. Restrictions on environments and target structures ($|E|, |E_T|, |X|$).

In Section 3.2.1, we discuss the general intractability of the *CoDesignLS* problem,

Table 3.1: Parameters for the controller-environment co-design problem.

Parameter	Description
$\|L\|$	# controllers in library
$\|T\|$	# robots in team
$\|Q\|$	# states per robot
d	# transitions per state
$\|f\|$	# symbols per transition-formula
r	Robot perceptual radius
$\|E\|$	# squares in environment
$\|E_T\|$	# distinct environment-square types
$\|X\|$	# squares in structure X

which leads to the need of parameterized complexity analysis. Hence, parameterized complexity results for parameters presented in Table 3.1 from previous work are described in Section 3.2.2 and parameterized results from new reductions are discussed later in Section 3.2.3. We discuss the implications of all of these results in Section 3.3. For the ease of the reader, we have given Table 3.2, which shows a summary of all the results derived in this chapter.

3.2.1 Classical Results from Previous Work

We consider first if controller-environment co-design under library selection can be done efficiently in general, i.e., if $CoDesignLS^{fast}_{syn/asy}$ is solvable in polynomial time and hence polynomial-time tractable. It turns out that this is not so by Lemma 8

Table 3.2: Summary of results for $CoDesignLS$.

Theorem	Description	Section
Theorem#1	Polynomial-time intractability of $CoDesignLS$	3.2.1
Theorem#3	$\langle \|L\|, \|T\|, \|X\|, \|Q\|, d\rangle$-$CoDesignLS^{fast}$ is fp-intractable for both *syn* and *asy*, when $\|L\| = \|T\| = \|X\| = \|Q\| = 1, d = 3$	3.2.2
Theorem#4	$\langle \|L\|, \|T\|, \|X\|, \|Q\|, \|f\|\rangle$-$CoDesignLS^{fast}$ is fp-intractable for both *syn* and *asy*, when $\|L\| = \|T\| = \|X\| = 1, \|Q\| = 5, \|f\| = 3$	3.2.3
Theorem#5	$\langle \|L\|, \|T\|, \|X\|, \|f\|, d\rangle$-$CoDesignLS^{fast}$ is fp-intractable for both *syn* and *asy*, when$\|L\| = \|T\| = \|X\| = 1, \|f\| = d = 3$	3.2.3
Theorem#6	$\langle \|X\|, \|Q\|, \|f\|, d\rangle$-$CoDesignLS^{fast}$ is fp-intractable for both *syn* and *asy*, when $\|Q\| = 4, \|f\| = 7, d = 3, \|X\| = 1$	3.2.3
Theorem#7	$\langle \|L\|, \|E\|, \|E_T\|\rangle$-CoDesignLS$^{fast}_{syn}$ is fp-tractable	3.2.3

in [37, Supplementary Material]. This lemma gives a reduction from Clique to the environment design problem (see below) introduced in [37]. This problem outputs an environment E for a given team T such that the team T creates the required structure X while working in environment E. Clique and the environment design problem are defined as follows:

Clique [15, 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 $u, v \in V'$, $(u, v) \in E'$? [2]

Environment Design ($EnvDes$)[37]

Input: A 2-d World-grid W, square-type set E_T, an FSR team T, based on controller c, a structure X, an initial positioning p_I of T in W, and position p_X of X in W.

Output: Output an environment E derived from W and E_T such that T started at p_I in E creates X at p_X, if such an environment E exists, otherwise output special symbol $\perp$.

Analogous to $CoDesignLS^{fast}$, $EnvDes^{fast}$ with superscript $fast$ indicates the instance of the problem in which robot team complete the construction task efficiently, which means the teams construction task is (c_1, c_2)-completable relative to some constants c_1 and c_2. $EnvDes^{fast}_{syn}$ and $EnvDes^{fast}_{asy}$ refers to the synchronous and asynchronous versions of the problem respectively.

It is useful to give the reduction from Lemma 8 in [37, Supplementary Material] in detail with figures because this reduction is referred to multiple times later in our thesis. Essentially, this reduction constructs a team consisting of a single robot which, in an environment encoding a candidate solution $V' \subseteq V$, $|V'| = k$, of Clique, checks if (1) V' consists of k distinct vertices and (2) there is an edge in G between each pair of vertices in V'; if so, a structure X is created at p_X. Following is the detailed description of this reduction.

[2]Note that both here and in all the proofs involving Clique, we have renamed the set of edges in graph G to E' to avoid confusion with the robot-team construction environment E.

Lemma 1 CLIQUE polynomial-time reduces to EnvDes^{fast} such that in the constructed instance of EnvDes^{fast}, $|T| = |X| = c_1 = 1$, $c_2 = 2$, $|f| = 3$, and $|Q|$, r, and $|E|$ are functions of k in the given instance of CLIQUE.

Proof: Given an instance $\langle G = (V, E'), k\rangle$ of CLIQUE, construct an instance $\langle W, E_T, T, X, p_I, p_X, c_1, c_2\rangle$ of EnvDes^{fast} as follows: Let W be a $2 \times k + 1$ grid, $E_T = \{e_0, e_1, \ldots e_{|V|}, e_F, e_X\}$ consist of $|V| + 3$ different types of free-space squares, $p_I = W_{1,1}$, and X be a single square of type e_X positioned at $p_X = W_{2,1}$. Team T will consist of a single FSR based on states $Q = \{q_0 = q_{U,0}, q_{U.1}, q_{U,2}. \ldots q_{U,k}, q_{E,1,1}, q_{E,1,2}, \ldots, q_{E,1,k}, q_{E,2,2}, q_{E,2,3}, \ldots, q_{E,2,k}, \ldots\ q_{E,k,k}, q_F, q_{Err}\}$, $|Q| = k + k(k-1)/2 + 3$, with the following transitions (state diagram for the controller of this single robot is given in Figure 3.4):

1. For each i, $1 \leq i \leq k$, the set of transitions $\{\langle q_{U,i-1}, enval(v_j, (0, i)) \wedge enval(v_j, (0, l)), *, stay, q_{Err}\rangle \mid 1 \leq j \leq |V|$ and $1 \leq$ l $<$ i$\}$ and transition $\langle q_{U,i-1}, *, *, stay, q_{U,i}\rangle$;

2. A transition $\langle q_{U,k}, *, *, stay, q_{E,1,1}\rangle$;

3. For each i, $1 \leq i < k$, the sets of transitions $\{\langle q_{E,i,j-1}, enval(e_u, (0, i)) \wedge enval(e_v, (0, j)), *, stay, q_{E,i,j}\rangle \mid i < j \leq k$ and (u, v) $\in$ E$'\}$ and $\{\langle q_{E,i,j-1}, enval(e_v, (0, i)) \wedge enval(e_u, (0, j)), *, stay, q_{E,i,j}\rangle \mid i < j \leq k$ and (u, v) $\in$ E$'\}$ and transition $\langle q_{E,i,k}, *, *, stay, q_{E,i+1,i+1}\rangle$; and

4. Transitions $\langle q_{E,k,k}, enval(e_0, (0, 0)), *, goEast, q_F\rangle$ and $\langle q_F,\ enval(e_F, (0, 0)), e_X, stay, q_F\rangle$.

Note that, such an instance of EnvDes^{fast} can be constructed in time polynomial in

terms of input size of an instance of CLIQUE. The robot starts with ensuring that no two square-types of the k squares $E_{1,2}$, $E_{1,3}, \ldots E_{1,k+1}$ are the same by comparing the environment types of each pair of square blocks and if any pair of square block has same environment type then the robot will change its state to q_{Err}, thus terminating the construction process. This is done via transitions in point 1 above. Then via transitions in point 3, the robot ensures that each pair of squares in these k squares has types that correspond to an edge $e \in E'$. Thus via transitions in point 1 and in point 3, robot ensures that these k squares encode a clique of size k in G.

To prove correctness we need to show that there exists a $V' \subseteq V$ such that V' is a clique of size k in G if and only if there exists an environment E derived from W and E_T such that T started at p_I in E creates X at p_X. We prove the "if" part of correctness by constructing an environment E (shown in Figure 3.5) as follows:

1. $E_{1,1}$ has square-type e_0;
2. $E_{1,j+1}$, $1 \leq j \leq k$, has square-type corresponding to the jth vertex in V';
3. $E_{2,1}$ has square-type e_F; and
4. All remaining environment-squares have square-type e_0.

Note that in such an environment E, the single clique checker robot in the team T will progress from p_I to p_X while creating the required structure X. It can be seen that the operation of this FSR in environment E is deterministic as at any point in the construction process, at most one transition of the robot is triggered. To complete the construction task, the robot must execute $k + k(k-1)/2 + 2$ transitions. As $k + k(k-1)/2 + 2 < 2k^2 + 2 < 4k^2 + 4 < (2k+2)^2 < c_1(|E| + |Q| + |f| + d)^{c_2}$ when

$c_1 = 1$ and $c_2 = 2$, this means that this construction task is (c_1, c_2)-completable in both the synchronous and asynchronous senses (as there is only one robot in the team so synchronous and asynchronous operations are same) with respect to team T and initial position p_I when $c_1 = 1$ and $c_2 = 2$.

Now we prove the "*only if*" part of correctness. Suppose that there is an environment E such that when the robot is started at initial position p_I, the task of constructing target structure X at position p_X is (c_1, c_2)-completable with respect to team T and initial position p_I. By the rules of FSR operations given Section 3.1, this robot operates deterministically. The way we have defined the robot's controller, it can only move right from $E_{1,1}$ to $E_{2,1}$ if it can change its state from q_0 to q_F. Keeping in mind that $E_{1,1}$ has square-type e_0, and $E_{2,1}$ has square-type e_F. However, observe that this change in robot's state from q_0 to q_F can only happen if environment-squares $E_{1,2}$ through $E_{1,k+1}$ in the west-most column of the environment E have square-types which correspond to a clique of size k in the graph G.

To complete this proof, note that in the constructed instance of EnvDes^{fast}, $|T| = |X| = c_1 = 1$, $c_2 = 2$, $|f| = 3$, $|Q| = k + k(k-1)/2 + 3$, $r = k$, and $|E| = 2k + 2$. ∎

Observe that this is also a reduction from Clique to $CoDesignLS^{fast}_{syn}$ and $CoDesignLS^{fast}_{asy}$ in which team size $|T| = 1$ and the library of controllers L has this controller as its only member. This proof is true for both synchronous and asynchronous team operation as there is only one robot in the team so synchronous and asynchronous operations are essentially the same. This yields the following.

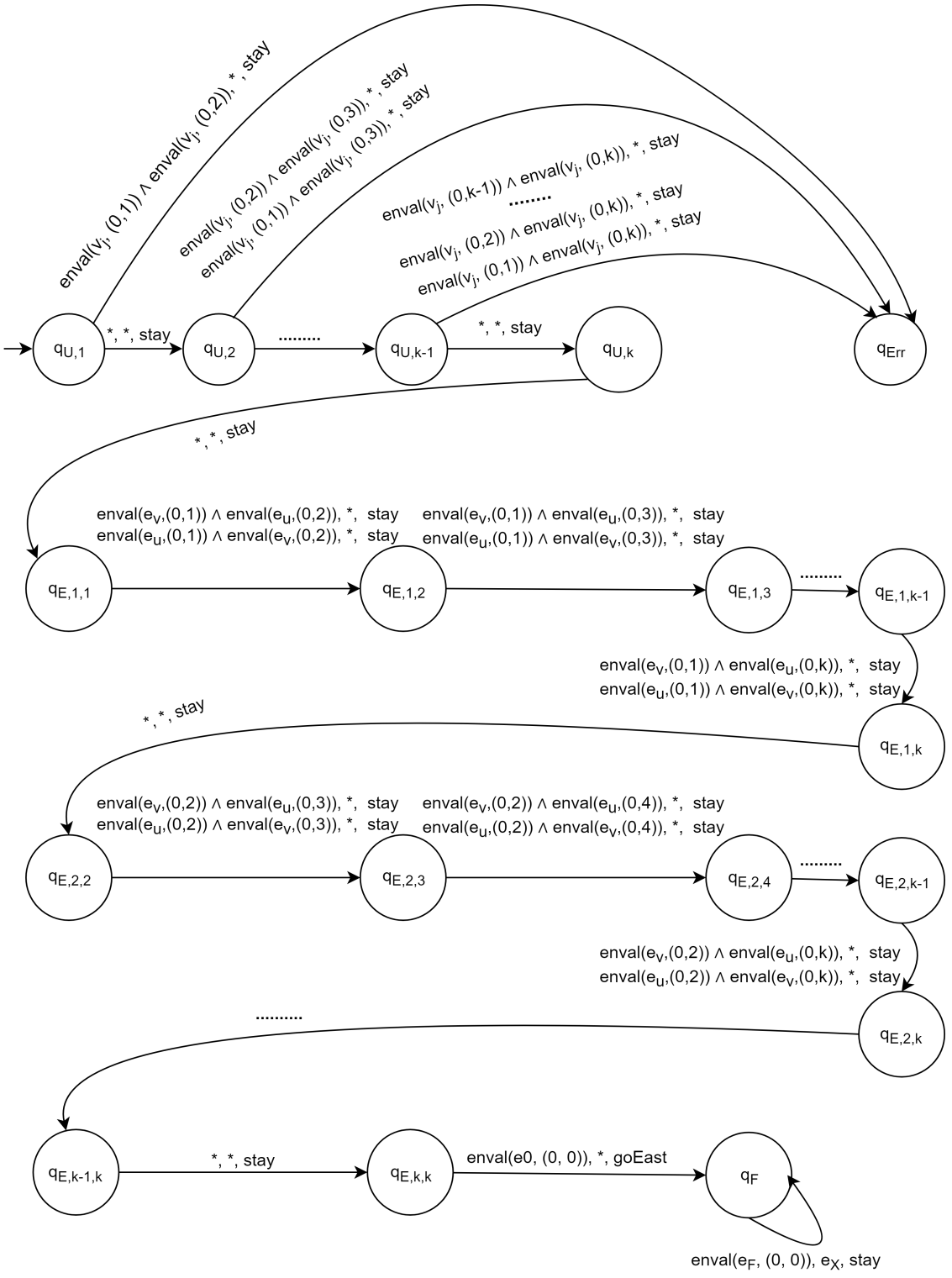

Figure 3.4: State Diagram of Clique checker robot used in Lemma 1. Note that in this diagram, $1 \leq j \leq |V|$ and e_v and e_u refer to the vertices of an edge (u, v) in E'.

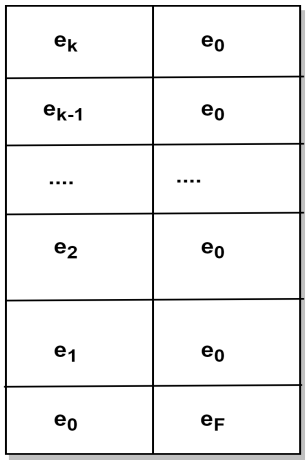

Figure 3.5: Environment used in the reduction in the proof of Lemma 1 and Theorems 3, 4 and 5.

Theorem 1 *If $P \neq NP$ then both $CoDesignLS_{syn}^{fast}$ and $CoDesignLS_{asy}^{fast}$ are not polynomial-time tractable.*

The NP-hardness of $CoDesignLS_{syn}^{fast}$ and $CoDesignLS_{asy}^{fast}$ has more impact than it initially seems. It also rules out the possibility of the existence of efficient probabilistic algorithms which operate correctly with probability $\geq 2/3$.

Theorem 2 *If $P \neq NP$ and $P = BPP$ then both $CoDesignLS_{syn}^{fast}$ and $CoDesignLS_{asy}^{fast}$ are not polynomial-time tractable by probabilistic algorithms which operate correctly with probability $\geq 2/3$.*

Proof: The proof is essentially the same as given in [37, Result D]. If either $CoDesignLS_{syn}^{fast}$ or $CoDesignLS_{asy}^{fast}$ has a polynomial time probabilistic algorithm that solves them with correctness $\geq 2/3$ then the decision versions[3] of these will also

[3]A decision version of problem is essentially the same problem with an answer of "yes" or "no".

have such probabilistic algorithms because an algorithm for a non-decision problem can be used to solve the decision version of the same problem [37, Lemma1, Supplementary Material]. This means that the decision version of $CoDesignLS^{fast}_{syn}$ and $CoDesignLS^{fast}_{asy}$ are in BPP as BPP is the most inclusive class of decision problems that can have probabilistic solutions particularly with a probability of correctness $\geq 2/3$. So if $P = BPP$, which is widely believed to be true [41, Section 5.2], and the decision versions of $CoDesignLS^{fast}_{syn}$ and $CoDesignLS^{fast}_{asy}$ are NP-hard (from Theorem 1[4]) then $P = NP$, completing the proof. ∎

Both of the results above in Theorem 1 and 2 are to be believed true if the conjectures $P \neq NP$ and $P = BPP$ are actually true.

The above rules out the possibility of general tractability for $CoDesignLS^{fast}_{syn}$ and $CoDesignLS^{fast}_{asy}$. Hence, we have to consider parameter combinations for these problems which if restricted, give fp-tractability. In Sections 3.2.2 and 3.2.3, we give multiple parameter sets which do not give fp-tractability after restricting to small constant values. We also give one combination of parameters that achieve fp-tractability for $CoDesignLS^{fast}_{syn}$.

3.2.2 Parameterized Results From Previous Work

The previous section showed that $CoDesignLS^{fast}_{syn}$ and $CoDesignLS^{fast}_{asy}$ are intractable in general. Now we will look into parameterized results for these problems to see if applying restrictions to different parameter sets of $CoDesignLS$ can yield tractability. We start with the reduction from Lemma 9 from [37, Supplementary

[4]A non-decision problem can be converted into decision version of the same problem by converting it into a question whether the required solution exists or not [37, Supplementary Material].

Material], which is a modification of the reduction from CLIQUE to $EnvDes$ given in Lemma 1. Here the number of states $|Q|$ and the number of transitions per state d are restricted at the expense of an increased maximum transition trigger-formula length $|f|$.

A brief description of this reduction is as follows. Given an instance $\langle G = (V, E'), k\rangle$ of CLIQUE, construct an instance $\langle W, E_T, T, X, p_I, p_X, c_1, c_2\rangle$ of EnvDes^{fast} as follows: Let W be a $2 \times k + 1$ grid, $E_T = \{e_0, e_1, \ldots e_{|V|}, e_F, e_X\}$ consist of $|V| + 3$ different types of free-space squares, $p_I = W_{1,1}$, and X be a single square of type e_X positioned at $p_X = W_{2,1}$. Team T will consist of a single FSR based on state $Q = q_0$, with $|Q| = 1$, with the following transitions:

1. $\langle q_0, f, *, goEast, q_0\rangle$. This one transition is obtained by merging multiple transitions from the proof of Lemma 1. The transition formula f is the conjunction of transition formulas of the following transitions from Lemma 1:

   (1) the negations of the non-default transition formulas in the transitions described in point 1 and

   (2) the non-default transition formulas of the transitions described in point 3 of the transition-list.

2. $\langle q_0, enval(e_F, (0, 0)), e_X, stay, q_0\rangle$

3. and the default transition $\langle q_0, *, *, stay, q_0\rangle$.

The proof of correctness of this reduction is essentially the same as that given in the proof of Lemma 1. The environment is same as used in Lemma 1 (see Figure 3.5) with a single robot with different controller as used in Lemma 1. As the robot

started at initial position p_I must execute 2 transitions to construct the structure X at p_X and $2 < (2k+2) < c_1(|E|+|Q|+|f|+d)^{c_2}$ when $c_1 = c_2 = 1$, this construction task is (c_1, c_2)-completable. Note that this reduction works for both the synchronous and asynchronous instances of $CoDesignLS^{fasy}$ with respect to team T and initial position p_I when $c_1 = c_2 = 1$ as there is only one robot in the team.

Observe that the reduction above is also a reduction from CLIQUE to $CoDesignLS^{fast}_{syn}$ and $CoDesignLS^{fast}_{asy}$ in which team size $|T| = 1$ and the library of controllers L has only one controller. This yields the following.

Theorem 3 *If $FPT \neq W[1]$ then both $\{|L|, |T|, |X|, |Q|, d, r\}$-$CoDesignLS^{fast}_{syn}$ and $\{|L|, |T|, |X|, |Q|, d, r\}$-$CoDesignLS^{fast}_{asy}$ are fp-intractable when $|L| = |T| = |X| = |Q| = c_1 = c_2 = 1$ and $d = 3$.*

In the following section we give parameterized intractability results for $CoDesignLS^{fast}$ using newly proved reductions.

3.2.3 New Parameterized Results

Using the same reduction technique as given in the proof of Lemma 1 in Section 3.2.1, we now derive additional results. These results follow from modifications to the reduction in the proof of Lemma 1. In the first of these modifications, we expand the number of transitions per state d while keeping the number of states $|Q|$ and maximum transition trigger-formula length $|f|$ constant.

Theorem 4 *If $FPT \neq W[1]$ then both $\{|L|, |T|, |X|, |Q|, |f|, r\}$-$CoDesignLS^{fast}_{syn}$ and $\{|L|, |T|, |X|, |Q|, |f|, r\}$-$CoDesignLS^{fast}_{asy}$ are fp-intractable when $|L| = |T| = |X| = 1$, $|Q| = 5$, and $|f| = 3$.*

Proof: Given an instance $\langle G = (V, E'), k\rangle$ of CLIQUE, construct an instance $\langle W, E_T, L, T, X, p_I, p_X, c_1, c_2\rangle$ of $CoDesignLS^{fast}$ as follows: Let W be a $2\times k+1$ grid, $E_T = \{e_0, e_1, \ldots e_{|V|}, e_F, e_X\}$ consist of $|V| + 3$ different types of free space squares, $p_I = W_{1,1}$, and X be a single square of type e_X positioned at $p_X = W_{2,1}$. Team T chosen from library L will consist of a single FSR c based on states $Q = \{q_0 = q_U, q_E, q_I, q_F, q_{Err}\}$, $|Q| = 5$, with the following transitions (state diagram given in Figure 3.6):

1. For each i, $1 < i \leq k$, the set of transitions $\{\langle q_U, enval(e_j, (0, i)) \wedge enval(e_j, (0, l)), *, stay, q_{Err}\rangle \mid 1 \leq j \leq |V|$ and $1 \leq$ l $<$ i$\}$ and transition $\langle q_U, *, *, stay, q_E\rangle$;

2. For each i, $1 < i \leq k$, the set of transitions $\{\langle q_E, enval(e_u, (0, i)) \wedge enval(e_v, (0, j)), *, stay, q_{Err}\rangle \mid 1 \leq j < i$ and (u, v) $\notin$ E$'\}$ and $\{\langle q_E,$ $enval(e_v, (0, i)) \wedge enval(e_u, (0, j)), *, stay, q_{Err}\rangle \mid 1 \leq j < i$ and (u, v) $\notin$ E$'\}$ and transition $\langle q_E, *, *, stay, q_I\rangle$; and

3. $\langle q_I, enval(e_0, (0, 0)), *, goEast, q_F\rangle$ and $\langle q_F,$ $enval(e_F, (0, 0)), e_X, stay, q_F\rangle$.

The robot starts with ensuring that no two square-types of the k squares $E_{1,2}$, $E_{1,3}, \ldots E_{1,k+1}$ are the same by comparing the environment types of each pair of square blocks and if any pair of square block has same environment type then the robot will change its state to q_{Err}, thus terminating the construction process. This is done via transitions in point 1 above. Then via transitions in point 2, the robot ensures that each pair of squares in these k squares has types that correspond to an edge $e \in E'$. Thus via transitions in point 1 and point 2, robot ensures that

these k squares encode a clique of size k in G. The state diagram for this clique checker controller used in the robot is given in Figure 3.6. Note that this instance of $CoDesignLS^{fast}$ can be constructed in time that is polynomial in the size of the given instance of CLIQUE.

To prove correctness we need to show that there exist a $V' \subseteq V$ such that V' is a clique of size k in G, if and only if there exist an environment E derived from W and E_T such that T started at p_I in E creates X at p_X. We prove the "if" part of correctness by constructing an environment E (shown in Figure 3.5) as follows:

1. $E_{1,1}$ has square-type e_0;
2. $E_{1,j+1}$, $1 \leq j \leq k$, has square-type corresponding to the jth vertex in V';
3. $E_{2,1}$ has square-type e_F; and
4. All remaining environment-squares have square-type e_0.

Note that in such an environment E, the single clique checker robot in the team T will progress from p_I to p_X while creating the required structure X. It can be seen that the operation of this FSR in environment E is deterministic in the sense described in Section 3.1. To complete the construction task, the robot c must execute 4 transitions (assuming if all other transitions going to q_{Err} do not activate). As $4 \leq c_1(|E| + |Q| + |f| + d)^{c_2}$ when $c_1 = c_2 = 1$, this means that this construction task is (c_1, c_2)-completable in both the synchronous and asynchronous senses with respect to team T and initial position p_I.

Now we prove the "*only if*" part of correctness. Suppose that there is an environment E such that when the robot is started at initial position p_I, the task of

constructing target structure X at position p_X is (c_1, c_2)-completable with respect to team T and initial position p_I. By the rules of FSR operations given in Section 3.1, this robot operates deterministically. The way we have defined the robot's controller, it can only move right from $E_{1,1}$ to $E_{2,1}$ if it can change its state from q_0 to q_F. Keeping in mind that $E_{1,1}$ has square-type e_0, and $E_{2,1}$ has square-type e_F. However, observe that this change in robot's state from q_0 to q_F can only happen if environment-squares $E_{1,2}$ through $E_{1,k+1}$ in the west-most column of the environment E have square-types which correspond to a clique of size k in the graph G.

Note that in the above constructed instance of $CoDesignLS^{fast}$, $|L| = |T| = |X| = c_1 = c_2 = 1$,$|f| = 3$, $|Q| = 5$, and $r = k$. The result then follows from the $W[1]$-hardness of k-Clique and the conjectured proper inclusion of FPT in $W[1]$. This proof is true for both synchronous and asynchronous team operation as there is only one robot in the team, so synchronous and asynchronous operations are essentially the same. ∎

In the second of these modifications, we expand the number of states while keeping the number of transitions per state and maximum transition trigger-formula length constant.

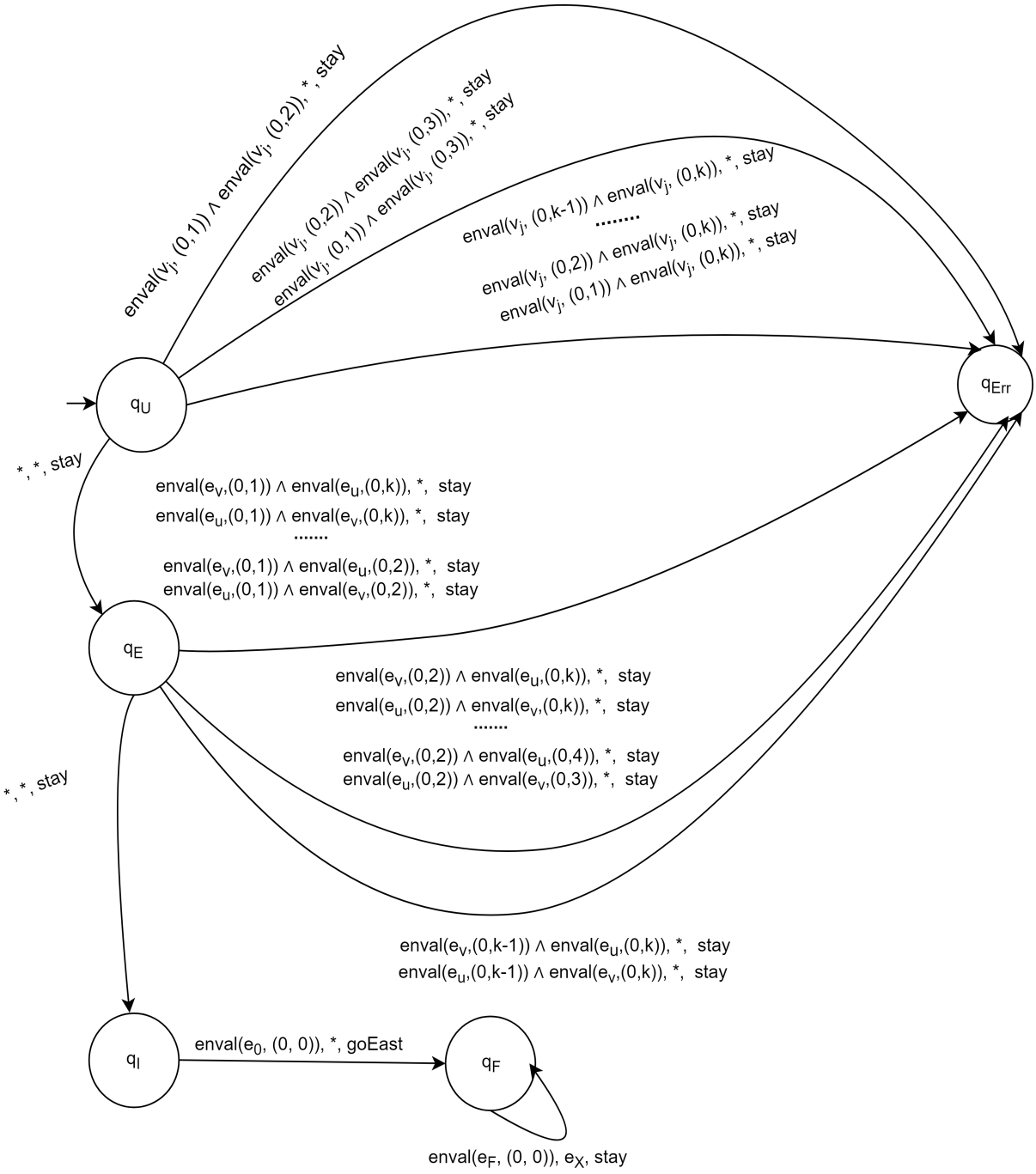

Figure 3.6: State Diagram of Clique checker robot used in Theorem 4. $1 \leq j \leq |V|$. e_u and e_v refer to the vertices u, v s.t $(u, v) \notin E'$.

Theorem 5 *If $FPT \neq W[1]$ then both $\{|L|, |T|, |X|, |f|, d, r\}$-$CoDesignLS^{fast}_{syn}$ and $\{|L|, |T|, |X|, |f|, d, r\}$-$CoDesignLS^{fast}_{asy}$ are fp-intractable when $|L| = |T| = |X| = 1$ and $|f| = d = 3$.*

Proof: The reduction below is obtained by modification (redistribution of transitions over an increased number of states) of proof given in Lemma 1. Given an instance $\langle G = (V, E'), k\rangle$ of CLIQUE, construct an instance $\langle W, E_T, L, T, X, p_I, p_X, c_1, c_2\rangle$ of $CoDesignLS^{fast}$ as follows: Let W, E_T, X, p_I, and p_X be as in the proof of Lemma 4, and team T consists of a single FSR c based on states $Q = \{q_0 = q_{U,1,1,1}, q_{U,1,1,2}, \ldots, q_{U,1,1,|V|+1}, q_{U,1,2,1}, \ldots, q_{U,1,2,|V|+1}, \ldots, q_{U,1,k,1}, q_{U,2,2,1}, \ldots, q_{U,2,2,|V|+1}, \ldots, q_{U,2,k,1}, q_{U,3,3,1}, \ldots q_{U,k,k,1}, q_{E,1,1,1}, q_{E,1,1,2}, \ldots, q_{E,1,1,|E'|+1}, q_{E,1,2,1}, \ldots, q_{E,1,2,|E'|+1}, \ldots, q_{E,1,k,1}, q_{E,2,2,1}, \ldots, q_{E,2,2,|E'|+1}, \ldots, q_{E,2,k,1}, q_{E,3,3,1}, \ldots q_{E,k,k,1}, q_{Err}, q_F\}$, $|Q| = k(k-1)(|E'| + |V| + 2)/2 + 2k + 2$, with the following transitions (see Figures 3.7 and 3.8):

1. For each i, $1 \leq i < k$, the set of transitions $\{\langle q_{U,i,j-1,m}, enval(e_m, (0, i)) \wedge enval(e_m, (0, j)), *, stay, q_{Err}\rangle, \langle q_{U,i,j-1,m}, *, *, stay, q_{U,i,j-1,m+1}\rangle, \langle q_{U,i,j-1,|V|+1}, *, *, stay, q_{U,i,j,1}\rangle \mid i < j \leq k$ and $1 \leq$ m $\leq$ |V|$\}$ and transition $\langle q_{U,i,k,1}, *, *, stay, q_{U,i+1,i+1,1}\rangle$ and transition $\langle q_{U,k,k,1}, *, *, stay, q_{E,1,1,1}\rangle$;

2. For each i, $1 \leq i < k$, the set of transitions $\{\langle q_{E,i,j-1,e}, enval(e_u, (0, i)) \wedge enval(e_v, (0, j)), *, stay, q_{E,i,j,1}\rangle, \langle q_{E,i,j-1,e}, enval(e_v, (0, i)) \wedge enval(e_u, (0, j)), *, stay, q_{E,i,j,1}\rangle, \langle q_{E,i,j-1,e}, *, *, stay, q_{E,i,j-1,e+1}\rangle, \langle q_{E,i,j-1,|E'|+1}, *, *, stay, q_{Err}\rangle \mid i < j \leq k$ and $1 \leq$ e $\leq$ |E′| and (u, v) $\in$ E′$\}$ and transition $\langle q_{E,i,k,1}, *, *, stay, q_{E,i+1,i+1,1}\rangle$; and transitions

3. $\langle q_{E,k,k,1}, enval(e_0, (0, 0)), *, goEast, q_F\rangle$, $\langle q_F, enval(e_F, (0, 0)), e_X, stay, q_F\rangle$.

The robot starts with ensuring that no two square-types of the k squares $E_{1,2}$, $E_{1,3}, \ldots E_{1,k+1}$ are the same by comparing the environment types of each pair of square blocks and if any pair of square block has same environment type then the robot will change its state to q_{Err}, thus terminating the construction process. This is done via transitions in point 1 above. Then via transitions in point 2, the robot ensure that each pair of squares in these k squares has types that correspond to an edge $e \in E'$. Thus via transitions in point 1 and point 2, robot ensures that these k squares encode a clique of size k in G.

The FSR has $|V|+1$ unique states for the comparison of each pair of square blocks type. It has $|E'| + 1$ separate states to check the existence of an edge between each pair of square blocks type. Note, this instance of $CoDesignLS^{fast}$ can be constructed in time that is polynomial in the size of the given instance of CLIQUE.

If there is a $V' \subseteq V$ such that V' is a clique of size k in G, construct an environment E (shown in Figure 3.5) as follows:

1. $E_{1,1}$ has square-type e_0;
2. $E_{1,j+1}$, $1 \leq j \leq k$, has square-type corresponding to the jth vertex in V';
3. $E_{2,1}$ has square-type e_F; and
4. All remaining environment-squares have square-type e_0.

Note that in such an environment E, the single clique checker robot in the team T will progress from p_I to p_X while creating the required structure X. It can be seen that the operation of this FSR in environment E is deterministic in the sense described in Section 3.1. To complete the construction task, c must execute at most

$k(k-1)(|E'|+|V|+1)/2+2k+1$ transitions which is $< |Q| < c_1(|E|+|Q|+|f|+d)^{c_2}$ when $c_1 = c_2 = 1$. This means that the construction task is (c_1, c_2)-completable in both the synchronous and asynchronous senses with respect to team T and initial position p_I when $c_1 = c_2 = 1$.

Conversely, suppose there is an environment E such that when c is started at p_I, the task of constructing X at p_X is (c_1, c_2)-completable with respect to T and p_I. This completability, by the rules of FSR operation given in Section 3.1, implies that the operation of c in E is deterministic. By the structure of this FSR, it can only move right from $E_{1,1}$ to $E_{2,1}$ if it can change state from q_0 to q_F, where $E_{1,1}$ has square-type e_0, and $E_{2,1}$ has square-type e_F. From any state $q_{U,i,j-1,m}$ FSR will go to q_{Err} if the square blocks at position $(0, i)$ and $(0, j)$ have the same square type e_m. Otherwise, by default, FSR will change its state and check the next environment type for the same square blocks until all environment types have been compared for the pair of square blocks at position $(0, i)$ and $(0, j)$. If the pair does not have a common environment type then FSR will compare the next pair and will repeat the process until all pairs of square blocks have been compared. If no pair has the same square type then the robot will change its state to $q_{E,1,1,1}$. From any state $q_{E,i,j-1,e}$ FSR will go to $q_{E,i,j,1}$ if the type of square blocks at position $(0, i)$ and $(0, j)$ corresponds to the vertices of eth edge. Otherwise, after checking the pair against each edge in E', it will change its state to q_{Err}. This thus ensures that the environment-squares $E_{1,2}$ through $E_{1,k+1}$ in the west-most column of E have square-types corresponding to a clique of size k in G.

Note, in the above constructed instance of $CoDesignLS^{fast}$, $|L| = |T| = |X| = c_1 = c_2 = 1$, $d = |f| = 3$, and $r = k$. The result then follows from the $W[1]$-hardness

of k-CLIQUE and the conjectured proper inclusion of FPT in $W[1]$. This proof is true for both synchronous and asynchronous team operation as there is only one robot in the team so synchronous and asynchronous operations are essentially the same. ∎

All of the proofs above use controllers with large numbers of states $|Q|$ or transitions per state d or long transition trigger-formulas $|f|$. It seems reasonable to conjecture that tractability will be achieved if we fix all of $|Q|$, d, and $|f|$ to small constant values. However, this proved not to be true by the next theorem.

Theorem 6 *If* $P \neq NP$ *then both* $\{|X|, |Q|, |f|, d\}$-$CoDesignLS^{fast}_{syn}$ *and* $\{|X|, |Q|, |f|, d\}$-$CoDesignLS^{fast}_{asy}$ *are fp-intractable when* $|Q| = 4$, $|f| = 7$, $d = 3$, *and* $|X| = 1$.

Proof: We obtain this result using a reduction from the following problem:

3-SATISFIABILITY ($3SAT$) [15, 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?

Given an instance $\langle U, C\rangle$ of $3SAT$, construct an instance $\langle E, E_T, L, T, X, p_I, p_X, c_1, c_2\rangle$ of $CoDesignLS^{fast}$ as follows: Let E be a $max(|C|, |U|) \times 3$ grid, $E_T = \{e_T, e_F, e_0, e_{c_1}, e_{c_2}, \ldots, e_{c_{|c|}}, e_{err}, e_X\}$ consisting of $|C| + 5$ different types of free space squares, p_I will be the first $|C|$ squares in the middle row, and X be a single square of type e_X positioned at $W_{1,1}$ (see Figure 3.9). Team T chosen from library L will consist of $|C|$ number of FSRs $c_1, c_2, \ldots, c_{|C|}$ based on states $Q = \{q_0, q_1, q_F, q_{Err}\}$, $|Q| = 4, d = 3$, with the following transitions (Figures 3.10, 3.11 and 3.12):

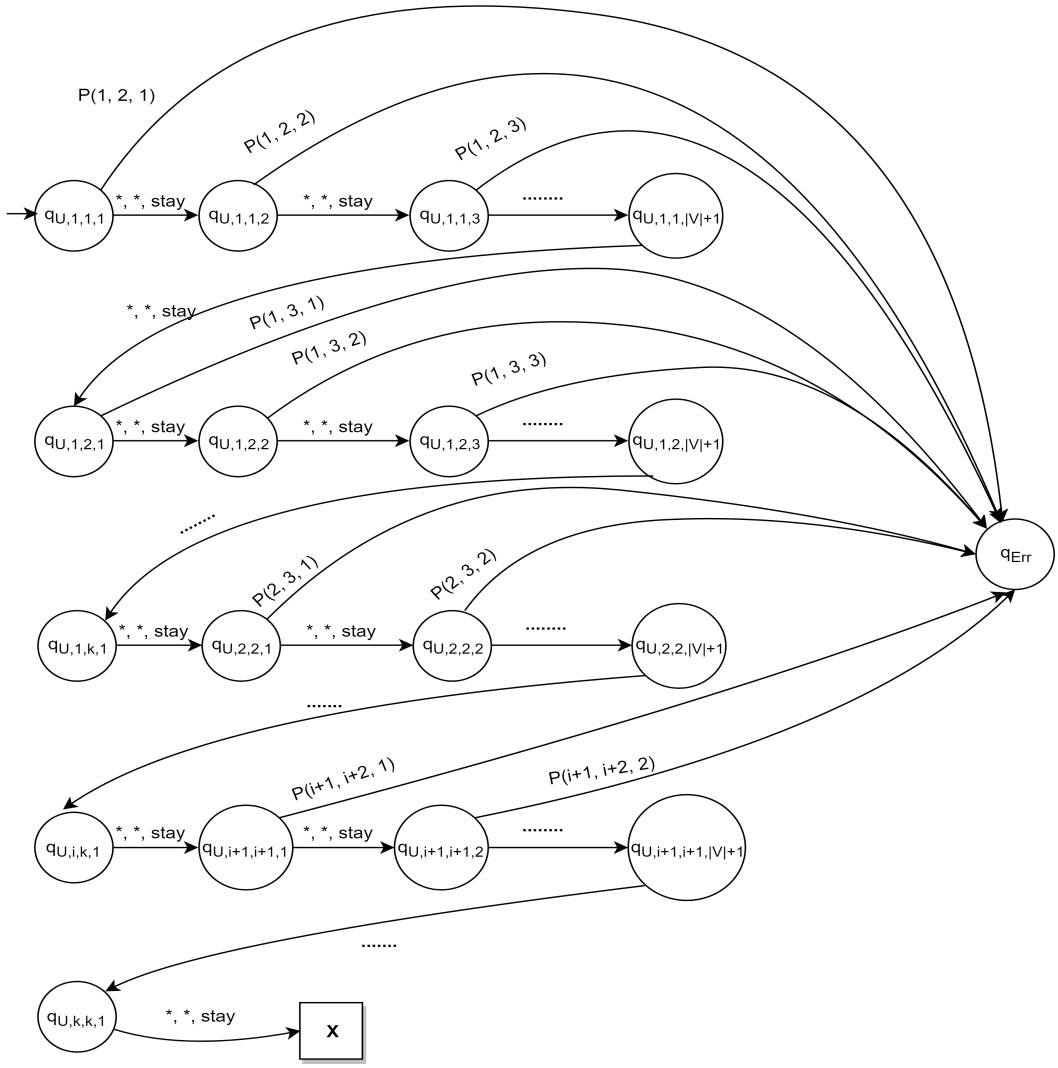

Figure 3.7: State diagram for checking uniqueness in Theorem 5. In this diagram, $P(i, j, m) = enval(e_m, (0, i)) \wedge enval(e_m, (0, j)), *, stay$

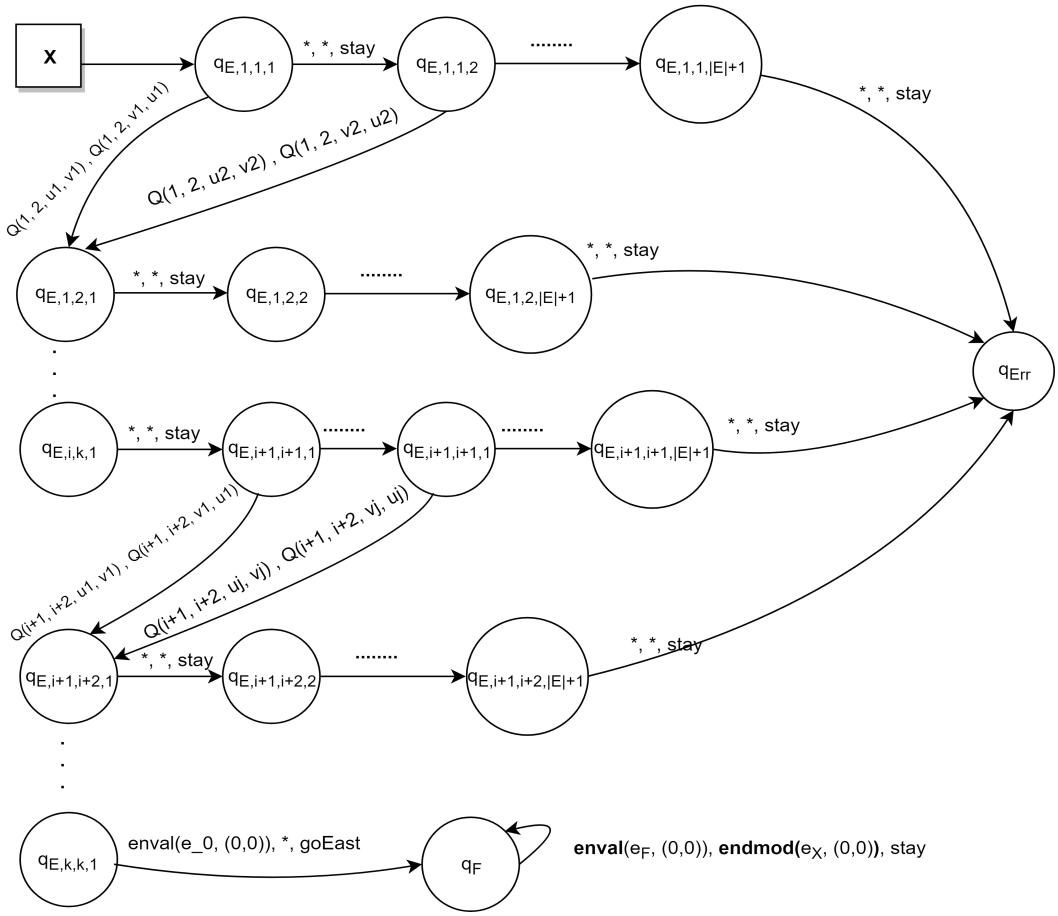

Figure 3.8: State diagram for edge check in Theorem 5. In this diagram, $Q(i, j, u, v) = enval(e_u, (0, i)) \wedge enval(e_v, (0, j)), *, stay$

1. Controller encoded with clause c_i, $1 \leq i \leq |C|$, has transition $\langle q_0, enval(e_{S_1}, (x,1)) \vee enval(e_{S_2}, (y,1)) \vee enval(e_{S_3}, (z,1)), *, stay, q_1\rangle$ where x, y and z are the x-coordinates of the position of the squares and $e_{S_1}, e_{S_2}, e_{S_3}$ represents the truth-value, either e_T (true) or e_F(false), corresponding to the three variables in clause c_i. And the transition $\langle q_0, *, enmod(e_{err}, (0,-1)), stay, q_{Err}\rangle$;

2. Controller representing clause c_i, where $2 \leq i \leq |C| - 1$ has the transitions $\langle q_1, enval(e_{c_{i-1}}, (-1,-1)) \wedge enval(e_{c_i}, (0,-1)) \wedge enval(e_{robot}, (1,-1)), *, goSouth, q_F\rangle$ and $\langle q_1, \neg enval(e_{c_{i-1}}, (-1,-1)) \vee \neg enval(e_{c_i}, (0,-1)) \vee enval(e_{err}, (1,-1)), enmod(e_{err}, (0,-1)), stay, q_{Err}\rangle$. Whereas for the controller representing clause c_1 the transitions are $\langle q_1, enval(e_{c_1}, (0,-1)) \wedge enval(e_{robot}, (1,-1)), *, goSouth, q_F\rangle$ and $\langle q_1, \neg enval(e_{c_1}, (0,-1)) \vee enval(e_{err}, (1,-1)), enmod(e_{err}, (0,-1)), stay, q_{Err}\rangle$. For the controller representing clause $c_{|C|}$ the transitions are $\langle q_1, enval(e_{c_{|U|-1}}, (-1,-1)) \wedge enval(e_{c_{|U|}}, (0,-1)), *, goSouth, q_F\rangle$ and $\langle q_1, \neg enval(e_{c_{|U|-1}}, (-1,-1)) \vee \neg enval(e_{c_{|U|}}, (0,-1)), enmod(e_{err}, (0,-1)), stay, q_{Err}\rangle$;

3. $\langle q_1, *, *, stay, q_1\rangle$ is for each controller;

4. This last transition is only for the controller representing clause c_1 $\langle q_F, enval(e_{c_1}, (0,0)), enmod(e_X, (0,0)), stay, q_F\rangle$

In the 1st transition, each robot, say r_i will sense the environment types of squares in the row above it, corresponding to three variables, x, y, z $(x, y, z \in U)$ in the clause c_i $(c_i \in C)$, encoded in it. r_i will check whether any of the three squares have an environment type matching the truth-value of the corresponding variables. Truth-

value of a variable is encoded in the ith controller as e_F if the complement of that variable is used in the ith clause otherwise e_T. If either of the three square types matches with the controllers' encoding, the robot will change its state to q_1. Here following actions are performed:

- Via the transitions in point 2 above, robot r_i will ensure that it has been placed in the correct position in the environment. It does this by making sure that the environment type of the square to its south is c_i which is placed in sequence with c_{i-1} to the left. This transition will be different for the controllers representing clauses c_1 and $c_{|C|}$. If the robot is placed in the correct sequence in the environment and it has ensured that the controller at its right has completed its job by moving to the south, it will change its state to q_F and move down south.

- r_i will change its state to q_{Err} if either the robot is not placed in the correct sequence or the robot to the right of r_i has changed the environment type of the square at position (1,-1), relative to r_i, to e_{err}.

Each robot will either change the square type of the square block to its south to e_{err} via transitions in point 4 or will move to the square block to its south. Note, in order for a robot to move to its south, it is necessary that all other robots to its right have moved down south (applying stigmergy [5]). If all the robots are placed in correct positions and all the clauses are evaluated to be true then each robot will move to the corresponding square block to the bottom row, and the robot encoding clause c_1 will change the type of square block at $E_{1,1}$ to e_X thus completing the construction task.

Note, this instance of $CoDesignLS^{fast}$ can be constructed in time that is polynomial in the size of the given instance of $3SAT$.

If there is a truth assignment for the variables in U which makes the set of clauses C in 3SAT evaluated to be true, construct an environment E as follows:

1. $E_{1,1}$ has square-type e_{c_1}, $E_{2,1}$ has square-type e_{c_2}, ..., $E_{|C|,1}$ has square-type $e_{c_{|C|}}$;

2. $E_{i,3}$ where $1 \leq i \leq |U|$, has square-type corresponding to the ith variable's truth-assignment in set U.

3. All remaining environment-squares have square-type e_0.

Observe that E will allow the robots in T to progress from p_I to create X at corresponding p_X; moreover, as at any point in this progress at most one transition in each FSR is enabled, the operation of robots is deterministic. To complete the task, r_1 will execute at max $2|C|$ transitions (as r_1 will wait for other robots to change their state to q_F). As $2|C| < 3(max(|C|, |U|)) < c_1(|E| + |Q| + |f| + d)^{c_2}$ when $c_1 = c_2 = 1$, this means that construction task is (c_1, c_2)-completable in both the synchronous and asynchronous senses with respect to T and p_I.

Conversely, suppose there is an environment E such that when each robot in the team is started at its corresponding p_I, the task of constructing X at p_X is (c_1, c_2)-completable with respect to T and p_I. This completability, by the rules of FSR operation given in Section 3.1, implies that the operation of robots in E is deterministic. By the structure of robots in T, each ith robot can move down south from $E_{i,2}$ to $E_{i,1}$ if it can change state from q_0 to q_F. However, this can only happen

if environment-squares $E_{1,3}$ through $E_{|U|,3}$ have square-types which correspond to a truth-assignment of variables in U that satisfies all clauses in C in the given instance of 3SAT.

Note, in the above constructed instance of $CoDesignLS^{fast}$,$|Q| = 4$, $|f| = 7$, $d = 3$, and $|X| = c_1 = c_2 = 1$. The result then follows from the NP-hardness of 3SAT, the conjectured proper inclusion of P in NP, and the observation that no problem parameterized relative to a parameter-set K can be in FPT if that problem is NP-hard when the values of all parameters in K are constants [37, Supplementary Materials, Lemma 2]. Observe that, the way controllers are designed, this proof works for both synchronous and asynchronous team operation, because in case of successful target structure creation, each robot will wait for the robot to its right to complete its execution and move down to south before it itself moves to the bottom row. ∎

These fp-intractability results have way more impact than it initially seems because if a problem is fp-intractable for a particular parameter-set K then it is fp-intractable also relative to any subset of K [33, Lemma 2.1.31]. Hence, almost none of the parameters considered can be restricted either individually to constant values or in combination to yield fp-tractability (see Table 3.3).

So far all our reductions have proved intractability for different parameter combinations for $CoDesignLS^{fast}$. However, all is not lost — there are sets of restrictions that do yield fp-tractability. One such set is as follows.

e_1	e_2	e_3	....	$e_{\|U\|-1}$	$e_{\|U\|}$	....	e_0	e_0
r_1	r_2	r_3	....	$r_{\|U\|-1}$	$r_{\|U\|}$	....	$r_{\|C\|-1}$	$r_{\|C\|}$
c_1	c_2	c_3	....	$c_{\|U\|-1}$	$c_{\|U\|}$	....	$c_{\|C\|-1}$	$c_{\|C\|}$

Figure 3.9: The environment used in Theorem 6. Each e_i, $1 \leq i \leq |U|$, represents the truth-value of the ith variable in U, where e_i is either e_T or e_F. Each r_i, $1 \leq i \leq |C|$, represents the initial position of clause robot i. In the lowermost row, the c_i, $1 \leq i \leq |C|$, are placed markers used by the clause robots to establish if they are in the correct position in the middle row.

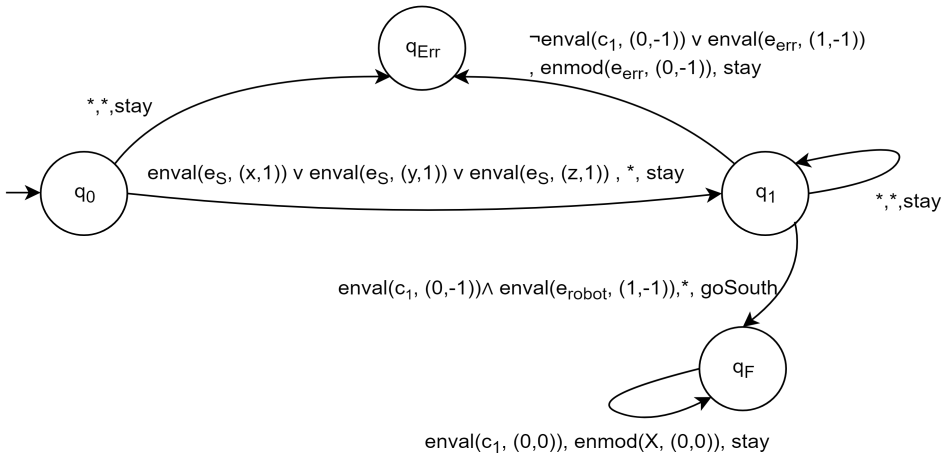

Figure 3.10: State diagram of the controller for the first clause robot used in Theorem 6.

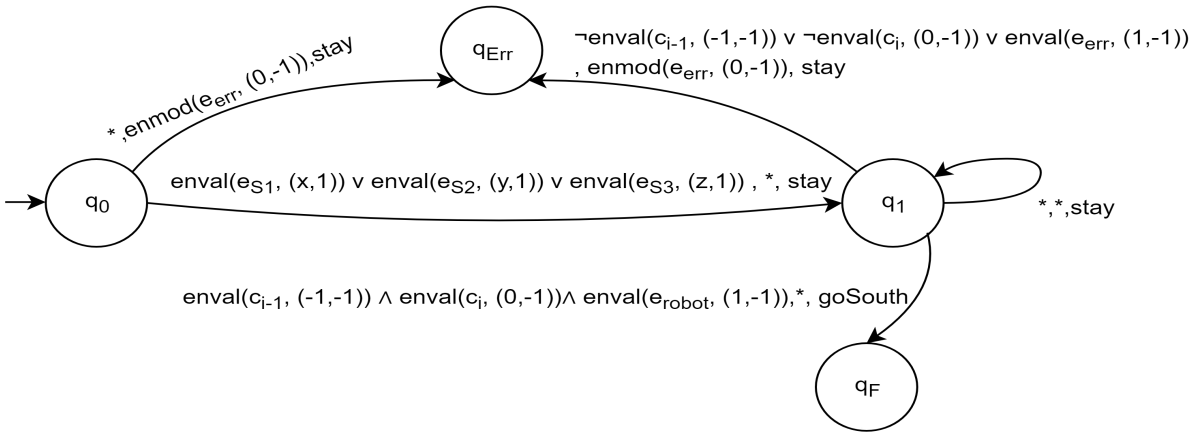

Figure 3.11: State diagram of the controller for the clause robot i, $2 \leq i \leq |C| - 1$, used in Theorem 6. Note that the controller for clause robot 1 given in Figure 3.10, which has the additional responsibility of placing requested structure X at $E_{1,1}$ if all clause robots move to the lowermost row, and for clause robot $|C|$ given in Figure 3.12 are modifications of the controller shown here.

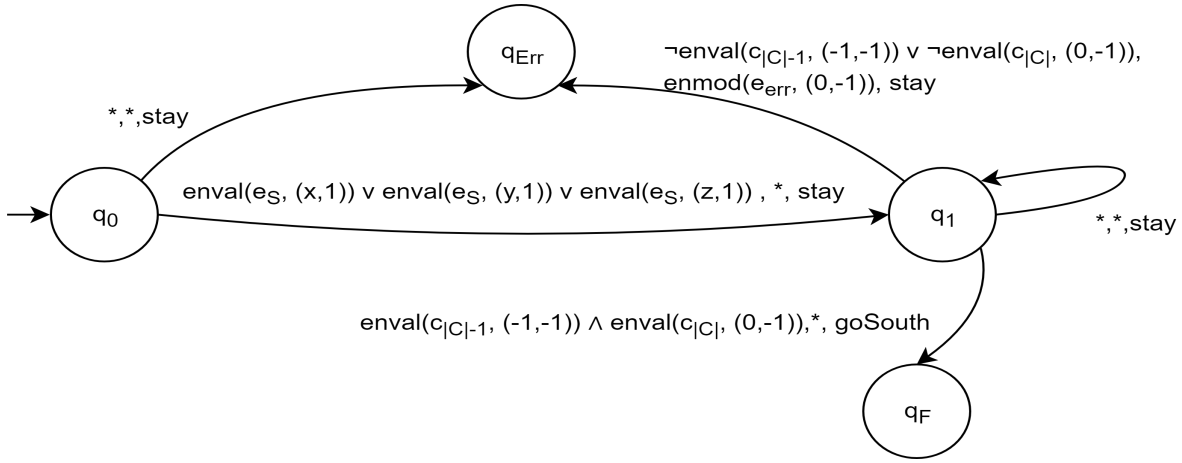

Figure 3.12: State diagram of the controller for the last clause robot used in Theorem 6.

Theorem 7 *$\langle |L|, |E|, |E_T| \rangle$-$CoDesignLS^{fast}_{syn}$ is fp-tractable.*

Proof: Consider the algorithm that generates all possible environments of size $|E|$ relative to E_T. For each such environment the algorithm will create all possible teams T of robots by selection from library L. Note that number of such possible teams is $|L|^{|T|}$. By placing T at all possible initial positions p_I, it determines for each team T, whether or not the created T started at p_I in that environment can create X at p_X in at most $c_1(|E| + |Q| + |f| + d)^{c_2}$ time-steps. The number of such environments is $|E_T|^{|E|}$.

The total number of combinations (say H) of robot teams, environments and placement of robots in the environment that the algorithm will verify for task completion will be $H \leq (|L|^{|T|})(|E_T|^{|E|})(|T|^{|p_I|})$. We know the following: $|T| \leq |E|$ (as at most one FSR from T can be present in each square of E) and $|p_I| \leq |E|$. Therefore $H \leq (|L|^{|E|})(|E_T|^{|E|})(|E|^{|E|})$.

Finally, the assessment of task completion in the required number of times-steps is done in FPT time. We know from [37, Supplementary Materials, Result L] that the verification of any team relative to a given environment can be done in FPT time relative to parameter-set $\{|E|, |E_T|\}$. As this verification considers all possible sequences of execution, it can be restricted to ensure that completion occurs in the required number of time-steps. Therefore the algorithm above shows that $\langle |L|, |E|, |E_T| \rangle$-$CoDesignLS^{fast}_{syn}$ is fp-tractable.

Note that this algorithm works only for synchronous team operation because team-environment verification algorithm in [37, Supplementary Materials, Result L] considers all non-repeating sequence of environment-configurations starting with the

members of T at p_I and ending with X at p_X. This non-repeating sequence of environment-configurations is only possible when a team of deterministic FSRs operates synchronously. ∎

3.3 Discussion

Theorem 1 shows that the controller-environment co-design through library selection ($CoDesignLS$) is polynomial-time intractable in general. In addition, Theorem 2 shows that there cannot exist any polynomial-time probabilistic algorithm that operates correctly more than two-thirds of the time. In other words, this means that at least one-third of the time, any polynomial-time probabilistic algorithm for controller-environment co-design will produce a robot team and environment which is not guaranteed to operate quickly, i.e., is $(c1, c2)-$completable in the sense defined in Section 3.1, and/or produce the requested structure at the requested position. This suggests that good evolutionary design algorithms may not be applicable to this problem.

Given the general intractability noted above, we considered what restrictions might get tractability. We derived a number of fixed-parameter intractability results. Theorems 3 - 6 show many parameter combinations relative to the parameters in Table 3.1 for which $CoDesignLS$ is fp-intractable. One interesting combination is given in Theorem 6, which restricts $|Q|, d$, and $|f|$ to constants to get fp-intractability. This means that even for the simplest form of the controller with a small constant number of states, number of transitions per state and transition-triggering formula length, $CoDesignLS$ is not solvable in FPT time. The intractability map in Table

3.3 shows that there are some combinations of parameters for which we were unable to show intractability results. Based on our failed attempts to create the reductions for the missing results, it seems that a small size team of robots with small values of $|Q|, d$, and $|f|$ has too little computational power to allow a reduction from an NP-hard problem. It can be seen that in Theorem 6, with small constant values of $|Q|, d$, and $|f|$, we had to increase the library size $|L|$ and the team size $|T|$ in order to sense and react properly in an environment encoding solution for $3SAT$ and to construct the target structure. It is our intuition that to accomplish a construction task relative to a reduction from an NP-hard problem with a small sized team, the robots in a team have to have either of the following: a sufficiently large set of states, a large number of transitions per state, or a long transition-triggering formula. If we are to use the simplest controllers with constant values of $|Q|, d$ and $|f|$, then we may have to have a sufficiently large number of robots in the team to complete the construction task efficiently relative to a reduction from an NP-hard problem. If the size of a team of robots is to be kept small and the library of controllers has the simplest controllers i.e. with a small value for $|Q|, d$ and $|f|$, one way to accomplish a construction task relative to a reduction from an NP-hard problem might be to allow the robots to have a transient memory. We investigate this option in Section 4.1.

Note that all of the intractability results given in Theorems 3 - 5 works for homogeneous team design as there is only one type of controller in the library. Theorem 6 is based on heterogeneous team design as there are different types of controllers in the library. It would be interesting to explore whether restricting $|Q|, d$, and $|f|$ relative to heterogeneous team design would still give intractability or not.

Based on our fp-intractability results in Theorems 3 - 6, it seems that in or-

der to get fp-tractability, restrictions on robot controllers may not be sufficient. As suggested in [37], along with small parameter values related to the controllers, restrictions on parameters related to the environment are needed to solve this problem efficiently. One such fp-tractability result is derived in Theorem 7. As the environment seems to be playing a key role here, we explore this problem further with respect to stigmergy-related parameters (which quantifies interaction between a robot and the environment) in Section 4.2.

Table 3.3: Summary of fp-intractability results for the controller-environment co-design problem. Results labelled X^{Rn} are the parameter-set results given in Table 3.2 for Theorems n ($n \in \{3,4,5,6\}$). Results labelled X^n are those derived by the subset-logic described after Theorem 6 in Section 3.2.3 from those given in Theorems n ($n \in \{3,4,5,6\}$). Results labelled ?? denote parameter-sets whose fp-status has not yet been established.

	—	$\|Q\|$	r	d	$\|Q\|, r$	$\|Q\|, d$	r, d	$\|Q\|, r, d$
—	NPh	X^4	X^4	X^5	X^4	X^3	X^5	X^3
$\|L\|$	X^4	X^4	X^4	X^5	X^4	X^3	X^5	X^3
$\|T\|$	X^4	X^4	X^4	X^5	X^4	X^3	X^5	X^3
$\|f\|$	X^4	X^4	X^4	X^5	X^4	X^6	X^5	X^{R6}
$\|L\|, \|T\|$	X^4	X^4	X^4	X^5	X^4	X^3	X^5	X^{R3}
$\|L\|, \|f\|$	X^4	X^4	X^4	X^5	X^4	??	X^5	??
$\|T\|, \|f\|$	X^4	X^4	X^4	X^5	X^4	??	X^5	??
$\|L\|, \|T\|, \|f\|$	X^4	X^4	X^4	X^5	X^{R4}	??	X^{R5}	??

Chapter 4

Controller-Environment Co-Design Through Library Selection: Elaborations

In this chapter, we discuss two elaborations on the basic model of controller-environment co-design given in Section 3.1. We will start with the introduction of a new controller architecture relative to which we have studied the *CoDesignLS* problem. Section 4.1 gives the motivation, description, intractability results and implication of the results for *CoDesignLS* relative to this new architecture. In Section 4.2 we talk about stigmergy and in Section 4.2.3 we give parameterized complexity results for stigmergic parameters for *CoDesignLS*.

4.1 Other Controller Architectures

In this section, we introduce a new controller architecture relative to which we have studied the $CoDesignLS$ problem. So far in the previous chapters, $CoDesignLS$ has been discussed under the SI model given in Section 3.1. For the remainder of this document, we will use the notation $CoDesignLS[SI]$ to refer to an instance of Controller-Environment Co-design under Library Selection relative to SI model and $CoDesignLS[New]$ will be used to refer to an instance of Controller-Environment Co-design under Library Selection relative to the model proposed in this section. Section 4.1.1 gives the motivation and description of the new controller architecture. Intractability result for $CoDesignLS[New]$ are given in Section 4.1.2 and implications of these results are discussed in Section 4.1.4.

4.1.1 Motivation

Here we perform a parameterized complexity analysis of the $CoDesignLS$ on a different controller architecture which is an extension of previously defined controller architecture of SI model in Section 3.1. The new controller architecture is designed by adding a transient memory to FSR while keeping rest of the controller architecture the same. As described in Section 3.1, each robot in the team can sense the environment type of a square-block within the Manhattan distance of their perceptual radius $r \geq 0$. Now with this new modification to robot's controller architecture, it will be able to temporarily memorize the environment type of the sensed square-block. Robots will use this memory to copy the environment type of one square-block to another square-block in the environment.

The inspiration of this transient memory came from the biological swarms of social insects e.g. swarm of ants and bees. Honey bees can memorize the scent of the flowers they visit [24] and ants use their memory to find food sources [11]. Introduction of transient memory to our robots is a better modeling of natural swarms and should lead us to many interesting complexity results.

In order to implement this transient memory we have introduced a new function and a new predicate. The new function $getEnVal(pos)$ returns the environment type of square-block at pos 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 $E_{i,j}$. For this new controller architecture, we do not replace predicate $enval(e, pos)$ defined in Section 3.1. However, unlike $enval(e, pos)$, which returned $true$ if the square-type of the square-block at position pos relative to robot's current position is e and $false$ otherwise, $getEnVal(pos)$ instead of giving $true$ or $false$ as a result, will return the environment type of square-block at position pos relative to robot's current position. A robot will memorize the sensed environment type and store that in temporary memory. Using that memory, the robot will recall what it just sensed and can copy that environment type to another square-block within the Manhattan distance $r \leq 1$. We can now use this newly defined function along with previously defined function $enmod(e, pos)$ to copy the environment type of one square-block positioned at $pos1$ relative to robot's current position to another square-block positioned at $pos2$ relative to robot's current position within the Manhattan distance $r \leq 1$ via $enmod(getEnVal(pos1), pos2)$. Note that $pos1$ and $pos2$ are defined in the same way as pos defined earlier.

The new predicate is $pairInSet((v_i, v_j), S)$, where S is a set of pairs e.g it can be

set of all pairs of vertices of a graph which have an edge between them. This predicate returns *true* if pair $(v_i, v_j) \in S$ and $false$ otherwise.

4.1.2 Main Result

In this section, we prove the fp-intractability result of $CoDesignLS[New]^{fast}$ relative to almost all of the parameters given in Table 3.1 by using only one reduction from CLIQUE.

Theorem 8 *If* $FPT \neq W[1]$ *then both* $\{|L|, |T|, |X|, |Q|, |f|, d, r\}$-$CoDesignLS[New]^{fast}_{syn}$ *and* $\{|L|, |T|, |X|, |Q|, |f|, d, r\}$-$CoDesignLS[New]^{fast}_{asy}$ *are fp-intractable when* $|L| = |T| = |X| = 1, |Q| = 8, |f| = 15$ *and* $d = 3$.

Proof: Given an instance $\langle G = (V, E'), k\rangle$ of CLIQUE, construct an instance $\langle W, E_T, L, T, X, p_I, p_X, c_1, c_2\rangle$ of $CoDesignLS[New]^{fast}$ as follows: Let W be a $2\times k+2$ grid, $E_T = \{e_0, e_1, \ldots e_{|V|}, e_T, e_F, e_X\}$ consist of $|V| + 4$ different types of free space squares, $p_I = W_{1,1}$, and X be a single square of type e_X positioned at $p_X = W_{2,1}$. Team T chosen from library L, will consist of a single FSR c based on states $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_F, q_{Err}\}$, $|Q| = 8, d = 3$, with the following transitions (see Figure 4.1 for the state diagram of the controller):

1. $\langle q_0, *, *, goNorth, q_1\rangle$;
2. $\langle q_1, *, enmod(getEnVal(0,0),(1,0)), goNorth, q_2\rangle$;
3. $\langle q_2, *, enmod(getEnVal(1,-1),(1,0)), goNorth, q_2\rangle$;
4. $\langle q_2, [enval(getEnVal(1,-1),(0,0)) \quad \vee \quad [\{\neg pairInSet((getEnVal(0,0),$

$getEnVal(1,-1)), E')\} \wedge \{\neg pairInSet((getEnVal(1,-1), getEnVal(0,0)), E')\}]] \wedge$

$\neg enval(e_T,(0,0)), *, stay, q_{Err}\rangle$;

5. $\langle q_2, enval(e_T,(0,0)), *, goSouth, q_3\rangle$;
6. $\langle q_3, *, *, goSouth, q_3\rangle$;
7. $\langle q_3, enval(getEnVal(1,0),(0,0)), *, goNorth, q_1\rangle$;
8. $\langle q_1, enval(e_T,(0,1)), *, goSouth, q_4\rangle$;
9. $\langle q_4, *, *, stay, q_{Err}\rangle$;
10. $\langle q_4, enval(e_0,(0,-k)), *, goSouth, q_5\rangle$;
11. $\langle q_5, *, *, goSouth, q_5\rangle$;
12. $\langle q_5, enval(e_0,(0,0)), *, goEast, q_F\rangle$;
13. $\langle q_F, enval(e_F,(0,0)), enmod(e_X,(0,0)), stay, q_F\rangle$;

Starting from p_I, robot will move up north to $E_{1,2}$. Now robot will sense the square-block's environment type underneath it and will compare it to other $k-1$ square-block's environment type by copying the environment type of sensed square-block to the east-most column. Example of such an environment is given at the end of the proof in Figure 4.2 (in this figure environment type of square-block $E_{1,2}$ is copied to $2nd$ column and being compared with environment type of other square-blocks). After performing the comparison for 1st square-block, the robot will choose next square-block and compare that with the remaining $k-2$ square-blocks. It will repeat this until all pairs of square-blocks have been compared.

The $2nd$ transition in the transition-list above copies the environment type of square-block underneath the robot to the square-block to the east of robot's current position. The $3rd$ transition copies the environment type of square to its south-east to the square block to its east and will move up north until following two cases:

- Robot reaches the north most square-block with environment type e_T (checked via transition 5). In this case, the robot will change state to state 3 and move down south. From state 3, it will keep moving down south until it reaches the square-block which has environment type same as the environment type of the square to its east, ensuring that it reaches the square block which the robot copied initially for the comparison. From here robot will move up north and choose the next square-block's environment type for comparison with other square-blocks environment type and change its state back to state 1.

- Robot finds two square-blocks with same environment type or two square-blocks do not have an edge between them (this is done via transition 4), in which case it goes to the error state.

This is done to make sure that no two square-types of the k squares $E_{1,2}, E_{1,3}, \ldots$ $E_{1,k+1}$ are the same and each pair of squares in these k squares has type that corresponds to vertices with edge $e \in E$; together, these transitions ensure that these k squares encode a clique of size k in G. Note that this instance of $CoDesignLS[New]^{fast}$ can be constructed in time polynomial in the size of the given instance of Clique.

To prove correctness we need to show that there exists a $V' \subseteq V$ such that V' is a clique of size k in G if and only if there exists an environment E derived from W and E_T such that T started at p_I in E creates X at p_X. We prove the "if" part of

correctness by constructing an environment E (shown in Figure 4.2) as follows:

1. $E_{1,1}$ has square-type e_0;
2. $E_{1,k+2}$ has square-type e_T;
3. $E_{1,j+1}$, $1 \leq j \leq k$, has square-type corresponding to the jth vertex in V';
4. $E_{2,1}$ has square-type e_F; and
5. All remaining environment-squares have square-type e_0.

Note that in such an environment E, the single clique checker robot in the team T will progress from p_I to p_X while creating the required structure X. It can be seen that the operation of this FSR in environment E is deterministic in the sense described in Section 3.1. To complete the construction task, the robot c must execute above 8 transitions $3k+k(k+1)+4$ number of times. As $(k^2+4k+4) < (2k+4)^2 = c_1|E|^{c_2} < c_1(|E|+|Q|+|f|+d)^{c_2}$ when $c_1 = 1$ and $c_2 = 2$, this means that this construction task is (c_1, c_2)-completable in both the synchronous and asynchronous senses with respect to team T and initial position p_I when $c_1 = 1$ and $c_2 = 2$. As there is only one robot in the team so synchronous and asynchronous operations are essentially the same.

Now we prove the "*only if*" part of correctness. Suppose that there is an environment E such that when the robot is started at initial position p_I, the task of constructing target structure X at position p_X is (c_1, c_2)-completable with respect to team T and initial position p_I. By the rules of FSR operations given Section 3.1, this robot operates deterministically. The way we have defined the robot's controller, it can only move right from $E_{1,1}$ to $E_{2,1}$ if it can change its state from q_0 to q_F. Keeping

in mind that $E_{1,1}$ has square-type e_0, and $E_{2,1}$ has square-type e_F. However, this can only happen if environment-squares $E_{1,2}$ through $E_{1,k+1}$ in the west-most column of E have square-types which correspond to a clique of size k in G. Because transition 4 makes sure that if any pair of vertices is the same or does not have an edge between them then FSR will change its state to error and will not create a structure. After making sure that all the pair of vertices are unique and each pair has an edge, transition 10 plays its role in assuring that the number of vertices it checked is k. When FSR is on square-block $E_{1,k+1}$, in order to ensure that it has checked k number of square block, robot senses the environment type of square-block at distance of k in the south. The environment type of this square should be e_0 ensuring that robot is k blocks far from p_I and if this square is out of the environment then *enval* predicate will return $false$, thus ensuring that structure will be created only when robot has checked k number of square-blocks corresponding to clique of size k in G.

To complete the proof, note that in the constructed instance of $CoDesignLS[New]^{fast}$, $|L| = |T| = |X| = 1, r = k, |Q| = 8, |f| = 15, d = 3, c_1 = 1, c_2 = 2$ and $|E| = 2k + 4$. ∎

It is interesting to note that just by introducing transient memory to the controller architecture, we are able to prove intractability of $CoDesignLS[New]^{fast}$ for all the parameter sets given in the intractability map in Table 4.1. This is in contrast with $CoDesignLS[SI]^{fast}$, where even with multiple reductions, we still are not able to complete the intractability map in Table 3.3. This shows the power of a transient memory in FSR. In the following section, we compare the relative powers of the two controller architectures.

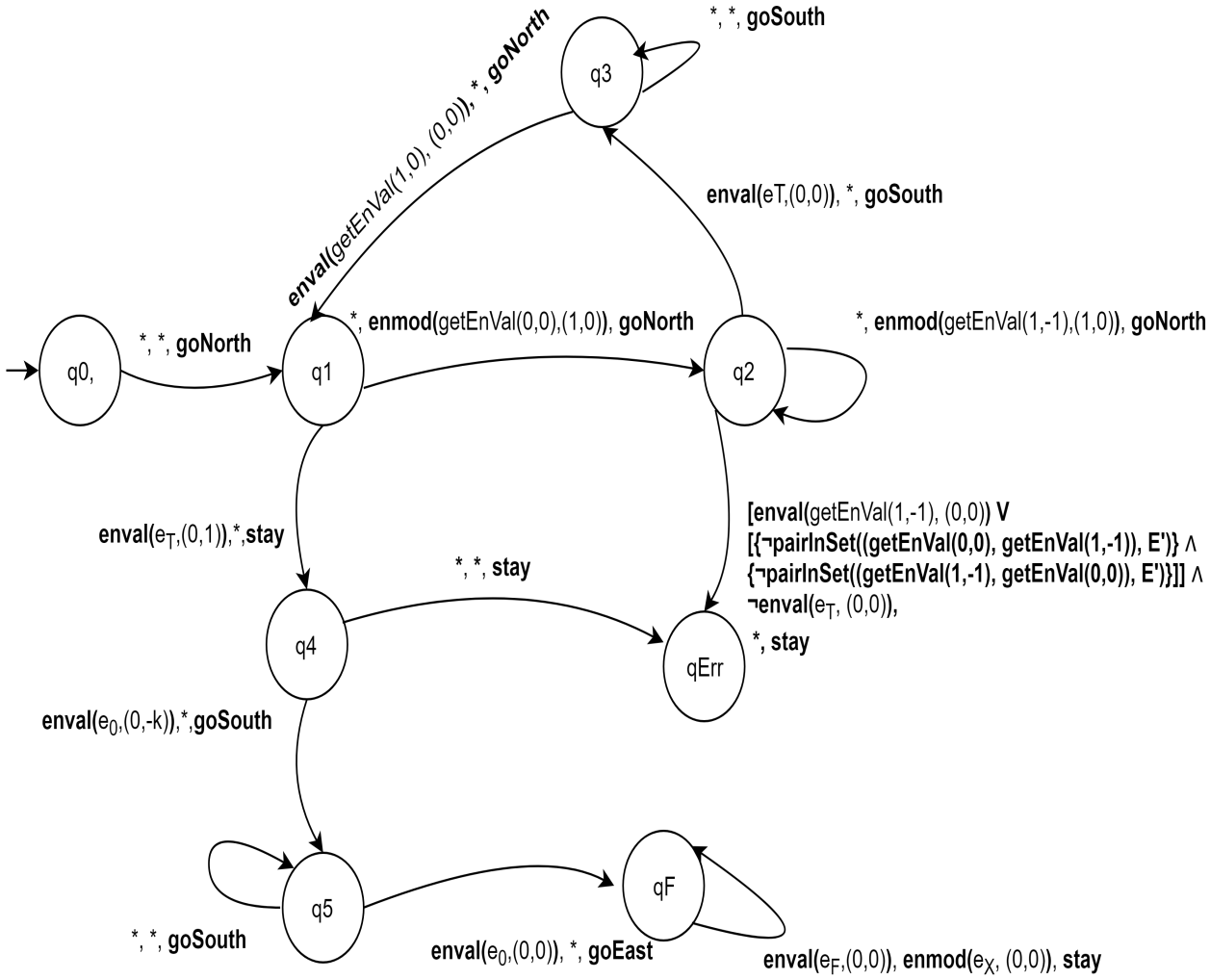

Figure 4.1: State diagram for controller used in proof of Theorem 8.

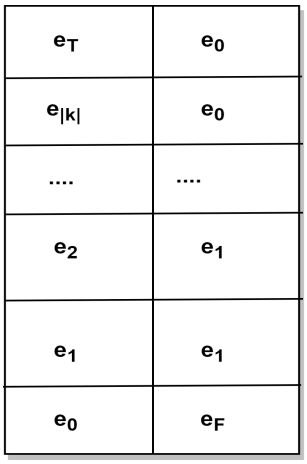

Figure 4.2: Environment used in proof of Theorem 8.

4.1.3 Reduction from SI to New Architecture

In order to compare the relative power of both architectures, we should see which one reduces to the other. It turns out that it is not possible to reduce from an instance of $CoDesignLS[New]$ to an instance of $CoDesignLS[SI]$ but fairly easy to reduce from an instance of $CoDesignLS[SI]$ to an instance of $CoDesignLS[New]$. To begin the reduction, we first consider the following new parameter e, which denotes the number of times memorization is used by any particular robot. It turns out that the difference between the two architectures can be quantified in terms of this new parameter. The following theorem shows the reduction from $CoDesignLS[SI]$ to $CoDesignLS[New]$.

Theorem 9 *$CoDesignLS[SI]$ polynomial-time reduces to $CoDesignLS[New]$ such that in the constructed instance of $CoDesignLS[New]$ all parameters are the same except $e = 0$.*

Proof: Given an instance of $CoDesignLS[SI]$, create an instance of $CoDesignLS[New]$ by assigning the parameters in $CoDesignLS[New]$ the same values as the values of the corresponding parameters in $CoDesignLS[SI]$ except the parameter e, which is set to 0. If there exists a solution for $CoDesignLS[SI]$ then it is easy to see that the solution would be the same for $CoDesignLS[New]$. There is no transient memory in $FSRs$ in the SI model and if a team of robots under SI model produces a target structure then the same team can produce the target structure under the *new* model, in which robots are allowed to have a transient memory. In short, if $FSRs$ in the new model are not allowed to use their transient memory i.e. by restricting $e = 0$, then their processing power is essentially the same as the $FSRs$ in the SI model. Hence, in this case, the solution for an instance of $CoDesignLS[SI]$ will also be the solution for instance of $CoDesignLS[New]$.

On the other hand, if there exist a solution for $CoDesignLS[New]$, the way we created the instance of $CoDesignLS[New]$ from the instance of $CoDesignLS[SI]$, $CoDesignLS[SI]$ will have the same solution by the logic given above (restricting $e = 0$ will make $CoDesignLS[New]$ essentially the same as $CoDesignLS[SI]$). This completes the proof. ∎

It is interesting to note that the fp-tractability result for the SI model given in Theorem 7 is also applicable to the new architecture.

Theorem 10 *$\langle |L|, |E|, |E_T| \rangle$-$CoDesignLS[New]^{fast}_{syn}$ is fp-tractable.*

Proof: The proof for this is essentially the same as the proof of Theorem 7 because all team operation rules given in Section 3.1 for the SI model are also applicable for the new architecture. So, the algorithm given in Theorem 7 that

verifies all possible combinations of robot teams, environments and placement of robots in the environment also work for the new architecture. Hence, $\langle |L|, |E|, |E_T| \rangle$-$CoDesignLS[New]^{fast}_{syn}$ is fp-tractable. ∎

4.1.4 Discussion

Recall that we had to have three reductions for $CoDesignLS[SI]$ to get even a partial intractability map in Table 4.1. On the other hand, adding transient memory allowed one reduction to give all the intractability results in that map when there is only one robot in the team i.e. a homogeneous team. This suggests that transient memory can replace the need for a large team with the simplest robots, i.e., with constant values of $|Q|, d$, and $|f|$, in building a reduction from an NP-hard problem. The new architecture is also clearly more powerful than the previous SI model as we only have the reduction from the SI to the new architecture, and we have not been able to derive a reduction in the other direction.

It is interesting to note that the fp-tractable result given in Theorem 7 for the SI model is also an fp-tractable result for the new architecture (as proved in Theorem 10). Along with a restriction on the controller's library size $|L|$, we also had to restrict parameters related to the environment ($|E|, |E_T|$) to get fp-tractability. This suggests that parameters related to the environment are crucial for fp-tractability results. Stigmergy is related to the interaction of robots with the environment. Analyzing $CoDesignLS$ relative to stigmergy-related parameters may give additional fp-tractability results. In the next section, we investigate stigmergy relative to $CoDesignLS$ using classical and parameterized complexity analyses.

Table 4.1: Summary of fp-intractability results for the controller-environment co-design problem relative to the new architecture. The result labelled X^{R8} is the full parameter-set result given in Theorem 8. Results labelled X^8 are those derived from the result given in Theorem 8 by the subset-logic described after Theorem 6 in Section 3.2.3.

	_	$\|Q\|$	r	d	$\|Q\|, r$	$\|Q\|, d$	r, d	$\|Q\|, r, d$
_	NPh	X^8	X^8	X^8	X^8	X^8	X^8	X^8
$\|L\|$	X^8	X^8	X^8	X^8	X^8	X^8	X^8	X^8
$\|T\|$	X^8	X^8	X^8	X^8	X^8	X^8	X^8	X^8
$\|f\|$	X^8	X^8	X^8	X^8	X^8	X^8	X^8	X^8
$\|L\|, \|T\|$	X^8	X^8	X^8	X^8	X^8	X^8	X^8	X^8
$\|L\|, \|f\|$	X^8	X^8	X^8	X^8	X^8	X^8	X^8	X^8
$\|T\|, \|f\|$	X^8	X^8	X^8	X^8	X^8	X^8	X^8	X^8
$\|L\|, \|T\|, \|f\|$	X^8	X^8	X^8	X^8	X^8	X^8	X^8	X^{R8}

4.2 Stigmergy

The self-organizing behavior of social insects has attracted the attention of scientists for a long time [17]. Social insects like ants, bees and termites do not seem capable of accomplishing a complex task when we look at an individual. However, working in a team they accomplish complex tasks e.g. mound building by termites. Research has shown that one of the major phenomenon lying behind the complex behavior of these biological swarms is stigmergy [5, 17]. Section 4.2.1 discusses the concept of stigmergy that exists in social insects. In Section 4.2.2 we quantify several stigmergy-

related parameters for $CoDesignLS$. In Section 4.2.3 we give fp-intractability results for $CoDesignLS$ relative to these stigmergy-related parameters and in Section 4.2.4 we discuss the implications of our fp-intractability results.

4.2.1 Motivation

The traces left in an environment by some work done by an agent which later motivates the sequence of actions to be performed in the environment by the same or some other agent is the principle of stigmergy [17]. The fact that insects perform complex tasks of finding a food source and building their nests has fascinated scientists for a very long time. This has lead scientists to search for the missing pieces of the puzzle, which turned out to be stigmergy. The notion of stigmergy was first introduced by the French entomologist Pierre-Paul Grassé in 1959 [16, 17] to explain this behavior of insects, where the insects without any central governing source perform complex tasks such as nest repairing and food finding.

Mound construction by termites is a beautiful example of this behavior. Mound construction is a result of the collective behavior of termites. In the beginning, termites randomly place mud particles. Termites have a tendency to drop mud particles to parts of environments where there is an abundance of mud i.e. where other termites have already placed mud. Heaps of mud start appearing which later becomes columns and eventually growing into tall termite mounds [16, 17]. Thus, termites communicate with each other simply by making changes to their environment.

Trail laying in the process of finding food by ants is another beautiful example of stigmergy. Explorer ants move out of the nest in search of food, leaving pheromones

behind. When a food source is found, the trail of pheromones left by explorer ants is used by other ants to reach to the food source [29].

For a long time, stigmergy was mainly focused on the study of the collective behavior of insects. Later scientists realized that stigmergy was underlying other complex systems as well such as self-organization of the microtubules (see [17] and references). The general ability to tackle complex problems exhibited by such self-organizing multi-agent collectives became known as swarm intelligence [5].

In our research, the concept of stigmergy has been applied to teams of simple robots which tends to perform a complex task by following the same mechanism of self-organization without any central governing system and direct communication, as done by swarms of social insects. There is a lot of research that has been done to simulate the self-organizing behavior of social insects [9] but up until now, there is no computational complexity work done on stigmergy. For the very first time, stigmergy will be studied under the light of complexity analysis tools for robot teams used for the construction of target structures. In the following section, we introduce some stigmergy-related parameters and motivate the need for those parameters.

4.2.2 Parameters of Interest

In this section, we quantify the stigmergic behavior of social insects in terms of problem parameters. In Table 4.2 we have defined some stigmergy-related parameters. We first discuss the motivation for these stigmergy-related parameters, and then in the next section, we derive fp-intractability results for these parameters relative to our $CoDesignLS$ problem.

In biological swarms, for example, a swarm of termite, creating their mound, an individual termite will work in a certain part of the environment. As millions of termites are involved in the construction process, individual termites cover only certain parts of the environment. This leads us to the conclusion that an individual termite moves to, senses and, hence, modifies only certain regions of the environment it is working in. In order to mimic this behavior, we introduce the parameter $|E_m|$ which is the number of square-blocks of a particular environment an individual robot can move to while working in a team during a construction process. As the robot will move only to a certain number of square blocks, therefore it can sense or read only a part of the environment which we quantify as $|E_r|$. The movement to a limited number of blocks restricts an individual robot to change only a certain number of square-blocks in the environment; hence, parameter $|E_s|$ which is the number of square-blocks an individual robot can change the square-type of is defined.

Finally, to mimic the memorizing capability of an individual insect in biological swarms, we introduce a parameter e, which is the number of times memorization is used by any particular robot. This completes our stigmergy-related parameter-list. This final parameter e has proved to be the difference between $CoDesignLS[SI]$ and $CoDesignLS[New]$ (see Section 4.1.3). Here we add the above-defined stigmergy-related parameters to the list of parameter classification given at the beginning of Section 3.2:

3. Restrictions on stigmergy-related characteristics of a robot e, $|E_r|$, $|E_s|$, $|E_m|$.

In the following section we give fp-intractability results relative to the stigmergy-related parameters discussed above. It is important to note that all the stigmergy-

Table 4.2: Stigmergy-related Parameters for the controller-environment co-design problem.

Parameter	Description
$\|E_r\|$	# of square blocks robot reads
$\|E_s\|$	# of square blocks robot writes to
$\|E_m\|$	# of square blocks robot moves to
e	# of times robot memorize a square-block type

related parameters given in Table 4.2 belong to both $CoDesignLS[SI]$ and $CoDesignLS[New]$ (recall from Section 4.1.3 that parameter e, which quantifies the transient memory, has value 0 for $CoDesignLS[SI]$).

4.2.3 Results

In our research, we have considered several stigmergy-related parameters which are given in Table 4.2. It turns out that both $CoDesignLS[SI]$ and $CoDesignLS[New]$ remain intractable for multiple combinations these parameters when their values have been restricted to constants. Both of the results here are derived directly from theorems in Chapter 3.

Theorem 11 *If* $P \neq NP$ *then both* $\{|E_s|, |E_m|, e, |L|, |T|, |X|, |Q|, d\}$-$CoDesignLS[SI]^{fast}$ *and* $\{|E_s|, |E_m|, e, |L|, |T|, |X|, |Q|, d\}$-$CoDesignLS[New]^{fast}$ *are fp-intractable when* $|E_s| = 1, |E_m| = 1, e = 0, |L| = |T| = X = |Q| = 1, d = 3$.

Proof: Observe that in the reduction in the proof of Theorem 3, the robot while completing the construction task only moves once in the whole construction process

to the square-block towards its east, which makes $|E_m| = 1$. Also, only in the final step of construction, the robot writes to a single square block, which makes $|E_s| = 1$. As there is no transient memory in FSR used in Theorem 3, therefore $e = 0$. This completes the proof. ∎

Theorem 12 *If* $P \neq NP$ *then both* $\{|E_s|, |E_m|, |E_r|, e, |X|, |Q|, f, d\}$-$CoDesignLS[SI]^{fast}$ *and* $\{|E_s|, |E_m|, |E_r|, e, |X|, |Q|, f, d\}$-$CoDesignLS[New]^{fast}$ *are fp-intractable when* $|E_s| = 1, |E_m| = 1, |E_r| = 5, |Q| = 4, |f| = 7, d = 3, X = 1$.

Proof: Observe that reduction in Theorem 6 is also a proof of this theorem as each robot while completing the construction task only moves once in the whole construction process to the square-block in the bottom row, which makes the $|E_m| = 1$. Also, in case of successful construction, as only in the final step of construction the left-most robot writes to a single square block, which makes $|E_s| = 1$. Each robot only reads 5 square-blocks at max (reads 3 square-blocks for 3 clause variables and reads 2 square blocks from the bottom row to ensure its correct placement in the environment) in the entire construction process; therefore $|E_r| = 5$. Also, as there is no transient memory in $FSRs$ used in Theorem 6, therefore $e = 0$. This completes the proof. ∎

It is important to note here that both the theorems above are true for both synchronous and asynchronous team operation because of Theorems 3 and 6 on which the above theorems are based work for both synchronous and asynchronous team operation. The results above have shown that even if we restrict individual robots to move, sense and change the only certain number of square blocks, $CoDesignLS$ still remains intractable relative to both the SI and new model.

4.2.4 Discussion

For the first time, we have quantified stigmergy-related attributes of swarm robotics in terms of parameters of construction-related problems and given the first complexity and parameterized complexity analysis of the controller-environment co-design problem relative to these parameters. Results here show that $CoDesignLS$ is fp-intractable relative to all stigmergy-related parameters given in Table 4.2 when restricted individually and in some certain combinations. In particular, Theorem 11 proves intractability relative to restricted values of three of the stigmergy-related parameters along with the library and team size and Theorem 12 shows that when all four stigmergy-related parameters are restricted to small values, we still have intractability if team size is increased. It seems that, as we discussed in Section 3.3, team size is crucial. Thus, it is worth determining the fp-status of $CoDesingLS$ with restricted values for all four stigmergy-related parameters along with a small team size, i.e. characterizing the fp-status of $\{|E_s|, |E_m|, |E_r|, e, |Q|, f, d, |T|\}$-$CoDesignLS[New]^{fast}$. It seems that considering the simplest stigmergy-related parameters was surprisingly ineffective in giving us tractability; therefore, we should in future research also consider other stigmergy-related parameters.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we have examined the controller -environment co-design through library selection problem from the point of view of computational complexity. We have given the first classical and parameterized complexity analyses for this problem. In these analyses, we have shown that $CoDesignLS$ is polynomial-time intractable in general both deterministically (Theorem 1) and probabilistically (Theorem 2). Furthermore, the problem remains fp-intractable under restrictions to a number of parameters given in Table 3.1, both individually and in various combinations (Theorems 3 - 6). We have also given the first set of parameters whose restrictions will guarantee fp-tractability (Theorem 7). A summary of these results is given in Table 3.2.

We then looked into two elaborations of $CoDesingLS$. We have first shown that $CoDesignLS$ remains both polynomial-time intractable and fp-intractable under a new architecture, in which robots have transient memory, relative to small values of

all of the parameter combinations given in Table 4.1 (Theorem 8). To compare the relative power of the SI with the new architecture we showed a reduction from the SI to the new architecture (Theorem 9). Also, as all the parameter combinations are fp-intractable relative to homogeneous teams under new architecture, which is not the case under SI model, it is apparent that transient memory, which is the difference between the two architectures, is much more powerful than it initially seemed to be. In the second of these elaborations, we have defined the stigmergy-related parameters given in Table 4.2 for the $CoDesingLS$ problem and derived fp-intractability results relative to these parameters (Theorems 11 and 12).

From the above, we conclude that controller-environment co-design using library selection is much more difficult than what the authors in [37] initially thought. In order to achieve tractability further analyses of the problem are required, either in form of exploration of new parameters for this problem or analyses of simplified variants of the problem.

5.2 Future Work

Relative to the future work, it would be interesting to characterize results for parameter combinations for which we have not proved fp-tractability or fp-intractability for $CoDesignLS$ (see Table 3.3). For example, fp-tractability results may be lurking under the parameter combinations $\{|L|, |T|, |Q|, |d|, |f|\}$ or $\{|L|, |T|, |Q|, |d|, |f|, |E_s|, |E_m|, |E_r|, e\}$ for $CoDesignLS$. A second direction for future work is to explore new parameters related to control mechanisms other than the parameters presented in Table 3.1 or additional stigmergy-related parameters other than those presented

in Table 4.2.

A final direction for future work would be to define and explore variants of the problem analyzed in this thesis. It appears that designing team of robots and their operating environment simultaneously is simply too difficult. An interesting variant of this problem that should be studied is a controller design with library selection relative to modification (along the lines proposed in [37]). Another promising variant is that given an environment, we are only allowed to change a limited number of squares to design the environment for a given team to work in.

Bibliography

[1] H. Ardiny, S. Witwicki, and F. Mondada. Are autonomous mobile robots able to take over construction? A review. *International Journal of Robotics: Theory and Applications*, 4(3):10–21, 2015.

[2] C. Balaguer and M. Abderrahim. Trends in robotics and automation in construction. In C. Balaguer and M. Abderrahim, editors, *Robotics and Automation in Construction*, pages 1–20. InTech, 2008.

[3] C. Balaguer and M. Abderrahim. Trends in Robotics and Automation in Construction. In *Robotics and Automation in Construction*. InTech, 2008.

[4] M. Beekman, G. A. Sword, and S. J. Simpson. Biological foundations of swarm intelligence. In *Swarm intelligence*, pages 3–41. Springer, 2008.

[5] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[6] 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.

[7] T. Brogardh. Present and future robot control development– An industrial perspective. *Annual Reviews in Control*, 31(1):69–79, 2007.

[8] R. A. Brooks and A. M. Flynn. Fast, cheap and out of control. *Journal of The British Interplanetary Society*, 42:478–485, 1989.

[9] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, E. Bonabeau, and G. Theraula. *Self-organization in Biological Systems*. Princeton University Press, 2003.

[10] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin. Effective leadership and decision-making in animal groups on the move. *Nature*, 433(7025):513–516, 2005.

[11] A. Dornhaus and N. R. Franks. Individual and collective cognition in ants and other insects (Hymenoptera: Formicidae). *Myrmecological News*, 11:215–226, 2008.

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

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

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

[15] M. Garey and D. Johnson. *Computers and Intractability*. WH Freeman, New York, 1979.

[16] P.-P. Grassé. La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes sociaux*, 6(1):41–80, 1959.

[17] F. Heylighen. Stigmergy as a universal coordination mechanism: Components, Varieties and Applications. In *Human Stigmergy: Theoretical Developments and New Applications*, New York, NY, USA, 2015. Springer.

[18] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, Reading, MA, 1979.

[19] A. Jevtić and D. Andina de la Fuente. Swarm intelligence and its applications in swarm robotics. In *Proceedings of the WSEAS international conferences : 6th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics*. WSEAS, 2007.

[20] A. R. Lanfranco, A. E. Castellanos, J. P. Desai, and W. C. Meyers. Robotic surgery: A current perspective. *Annals of Surgery*, 239(1):14, 2004.

[21] T. Luettel, M. Himmelsbach, and H.-J. Wuensche. Autonomous ground vehicles-concepts and a path to the future. *Proceedings of the IEEE*, 100:1831–1839, 2012.

[22] S. V. Mammen, C. Jacob, and G. Kokai. Evolving swarms that build 3D structures. *IEEE Congress on Evolutionary Computation (CEC)*, pages 1434–1441, 2005.

[23] L. E. Parker and J. V. Draper. Robotics applications in maintenance and repair. In S. Nof, editor, *Handbook of Industrial Robotics*, pages 1023–1036. Wiley, 1998.

[24] L. J. Rogers and G. Vallortigara. From antenna to antenna: Lateral shift of olfactory memory recall by honeybees. *PLoS One*, 3(6):e2340, 2008.

[25] E. Sahin. Swarm robotics: From sources of inspiration to domains of application. In *International Workshop on Swarm Robotics*, pages 10–20. Springer, Berlin, 2004.

[26] K. S. Saidi, T. Bock, and C. Georgoulas. Robotics in construction. In *Handbook of Robotics*, pages 1493–1520. Springer, Berlin, 2016.

[27] T. Soleymani, V. Trianni, M. Bonani, F. Mondada, and M. Dorigo. Bio-inspired construction with mobile robots and compliant pockets. *Robotics and Autonomous Systems*, 74:340–350, 2015.

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

[29] D. J. Sumpter and M. Beekman. From nonlinearity to optimality: Pheromone trail foraging by ants. *Animal Behaviour*, 66(2):273–280, 2003.

[30] J. Tautz and K. Rohrseitz. What attracts honeybees to a waggle dancer? *Journal of Comparative Physiology A*, 183(5):661–667, 1998.

[31] G. Theraulaz and E. Bonabeau. Coordination in distributed building. *Science*, 269(5224):686–688, 1995.

[32] M. Timmar and T. Wareham. The computational complexity of controller-environment co-design using library selection for distributed construction. In *Springer Proceedings on Advanced Robotics Systems (SPAR) Series*. Springer Verlag, Heidelberg, 2018.

[33] T. Wareham. *Systematic Parameterized Complexity Analysis in Computational Phonology*. PhD thesis, University of Victoria, Canada, 1999.

[34] T. Wareham. 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*, pages 295–302, Brussels, 2015. ICST.

[35] T. Wareham. Designing teams of autonomous robots for distributed construction, repair and maintenance: A computational complexity perspective. *ACM Transactions on Autonomous and Adaptive Systems*, Submitted.

[36] T. Wareham, J. Kwisthout, P. Haselager, and I. van Rooij. 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*, volume 2, pages 1–5, 2011.

[37] T. Wareham and A. Vardy. Putting it together: The computational complexity of designing robot controllers and environments for distributed construction. *Swarm Intelligence*, 12(2):111–128, 2018.

[38] T. Wareham and A. Vardy. Viable algorithmic options for designing reactive robot swarms. *ACM Transactions on Autonomous and Adaptive Systems*, 13(1):5, 2018.

[39] J. Werfel and R. Nagpal. Three-dimensional construction with mobile robots and modular blocks. *The International Journal of Robotics Research*, 27(3–4):463–479, 2008.

[40] J. Werfel, K. Petersen, and R. Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014.

[41] 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.

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