INSTITUTO SUPERIOR TÉCNICO

From Bio-Inspired to Institutional-Inspired Collective Robotics

PTDC/EEA-CRO/104658/2008

Second milestone report

Danesh Tarapore, José Nuno Pereira, Porfirio Silva,
Anders Lyhne Christensen, Jorge Carneiro and Pedro Lima

28 February 2012

FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

Contents

Chapter 1

Introduction

The project "From Bio-Inspired to Institutional-Inspired Collective Robotics", aims to investigate and formalise laws that govern large-scale collective systems in order to synthesise systems consisting of relatively simple robots that display complex behaviour through local interactions. In order to achieve this endeavour, we have focused on both biological and social systems. From biology, we model a multi-cellular system, namely the T helper cells of the adaptive immune system. While from sociology, we have focused on institutional economics. Our objective is to bring together theories, ideas and inspiration from institutional economics and cell biology under a common formal framework for large robot populations modelling and analysis.

1.1 Challenges

The field of collective robotics has made rapid progress since its advent in the 1980s, with groups of robots being made to do a wide variety of tasks ranging from foraging and nest-construction (e.g. (Waibel et al., 2009; Parker et al., 2003)), to self-organisation and exploration (e.g. (Hauert et al., 2009; O'Grady et al., 2010; Stirling et al., 2010)). However despite these efforts, the existing task scenarios are largely limited to toy problems with few robots, and in carefully controlled laboratory environments. In dealing with large-scale robot collectives, a direct approach in programming individual robots may be insufficient. This is because it is difficult to predict beforehand the nature of the interactions between robots, and their effect on the collective task. Automated, self optimisation approaches have been widely used to avoid this problem (e.g. artificial evolution, (Waibel et al., 2009) and machine learning (Panait and Luke, 2005)). However, the results of these methods are difficult to analyse. Additionally, when dealing with large number of robots in a group, optimisation techniques are prohibitively expensive in computation.

The process of developing controllers for individual robots (micro-level) needs to be supported with models to design and robustly predict state of system (macro-level). Models allow designers of multi-robot systems to capture fundamental dynamics of these nonlinear, asynchronous, large-scale systems at more abstract levels, possibly achieving mathematical tractability. Modelling is also a means for enabling generalisation to different robotic platforms and estimating optimal system parameters, including control parameters. Furthermore, by developing models of the system, we can verify macro-level properties like stability, robustness, adaption and innovation.

A multi-level probabilistic modelling methodology for distributed robotic systems was proposed in (Martinoli et al., 2004). This methodology also takes into account the design of controllers for robots. Being a bottom-up approach, it starts by considering a real implementation of the system (or faithful simulations) and then builds up a series of increasingly abstract models, carefully validating each against those at lower abstraction levels. The methodology has been applied successfully to a large variety of distributed robotic systems (Winfield et al., 2008; Mermoud et al., 2010; Correll and Martinoli, 2011).

However, in some cases, this approach yields macroscopic models that are difficult to analyse mathematically (e.g., non-linear, time-delayed systems of differential or difference equations, sometimes partial). Being primarily designed for swarm robotics systems, the methodology might not be able to deal with the higher behavioural complexity we envision for institutional robotics applications. While design of an individual controller is still feasible using, for instance, a Finite State Machine (FSM) approach, modelling different behaviours encapsulated in one controller using this approach might lead to state explosion and a prohibitively large model.

1.2 Approach

1.2.1 Bio-inspired approach

The adaptive immune response is orchestrated by helper T (Th) lymphocytes that take critical collective decisions based on local information alone. These decisions are critical since they can imply either fighting or accepting invading microorganisms; engaging or not in autoimmune pathologies. The cells as a population dynamically regulate and differentiate themselves into different sub-lineages (e.g. Th1, Th2, Th17, and Treg cells) (Harrington et al., 2005; Hori et al., 2003; Park et al., 2005; Mosmann and Coffman, 1989) to initiate the appropriate immune response. Importantly, the differentiation of naive Th cells into the different functionally distinct sub-types is decentralised, based solely on the state of the cell and local information available to it.

Considerable research has been done on the genetic regulatory networks (GRN) controlling individual T helper cell differentiation (Naldi et al., 2010; Garg et al., 2009; Mendoza, 2006; Garg et al., 2007). However, an important consideration of most of these models is that an individual Th cell is analysed independent of its environment. Consequently, interesting properties of the cell population may not be characterised with such models. To this end, we have developed a detailed stochastic simulator of the Th cell collective, simulating the molecular reactions of each Th cell (guided by their GRN), interactions between cells, and population growth (Tarapore and Carneiro, 2010). The detailed nature of our simulator, coupled with the striking resemblance between cells and robots, allows the simulator to be used as a test bed for theoretical models of collective systems. Consequently, not only can the models be greatly improved to better design robot collectives, but they may also help in enriching the theory of immune response.

A fundamental difference between cells and robots, that hinders the direct usage of immune system inspired algorithms for collective robotics, is that robots cannot proliferate (except some modular robotic systems in which composite entities may replicate when sufficient individual units are available (Zykov et al., 2005)). In our approach, we also address this problem and suggest possible alternative to cell proliferation. Furthermore, we show analytically and with stochastic simulations, the properties of the system are retained.

1.2.2 Institutional inspired approach

In institutional robotics the agents are seen in a somewhat different light. While in population of cells, an individual agent is very reactive, following a simple set of rules at the local level, in human societies an agent is more deliberative, their behaviour being harder to capture. Institutional Robotics aims to consider agents as more complex then reactive agents, by considering that through cooperative decision-making processes, agents can actually modify the rules that guide their behaviour. Coordination between agents is achieved by this self-regulation of social interactions since the robots know not only how to behave in a given scenario but also what to expect from other robots and the environment.

According to the IR approach:

1. the coordination system is a network of institutions;

2. institutions are coordination artifacts of different types (organisations, teams, hierarchies, conventions, norms, social roles, behavioural routines, stereotyped ways of sensing and interpret certain situations, material devices and particular organisations of the physical world), that may be implemented as material objects and/or mental constructs;

3. robots are able to modify, at some extent, the material Organization of its physic environment (and so modify the material basis of the institutions);

4. robots are able to deliberately modify the institutional environment: “institutional imagination” (“thought experiences” about possible outcomes of modifying current institutions) and “institutional building” (collective decision-making processes to modify “constitutional rules” of current institutions) are mechanisms to do so;

5. there are non-deliberate means of institutional evolution: institutions can be modified by accumulating small modifications initiated by some robots and not opposed by others; item robots with institutional building capabilities need a high degree of autonomy, being able to pursue their own goals grounded on their “struggle for survival” (some form of homoeostasis for artificial agents).

Summarising, from an institutional perspective, institutions are taken as the main tool of any sophisticated society, and individuals are both constructive within and constructed through institutional environments. With this heuristic, the institutional approach to multi-robot systems may provide appropriate conceptual means to the current methodological needs of designing complex coordination algorithms for distributed robotic systems.

1.2.3 Mathematical models

The mathematical model (see Chapter 2) may not scale up to more complex systems involving a higher number of micro-level states, the possibility of adding new states, and scenarios with more complex (heterogeneous) environments. In such cases, as (i) detailed multi-cellular stochastic simulations, or (ii) realistic collective robotic scenarios, e.g., the novelty detection case discussed earlier, another mathematical formalism is needed. A stochastic petri net formalism (bottom-up approach) is presented (see Chapter 4), that may be more adequate to scale up with more complex simulations.

With the resulting model, stability can be studied by analysing the impact of eliminating state transitions corresponding to communication among collective members. Modelling and analysing short-term adaptations refer to environment parameter modifications over time, which can be learnt, e.g., through reinforcement learning. Innovation implies changing the structure of the PN or SHA modelling the collective behaviour and analysing the impact of such drastic modifications. Robustness is tested by adding or removing sub-PN or sub-SHA corresponding to models of sub-group roles and re-computing the full robot- collective/environment closed loop model.

The information about the state of a given system being modelled is distributed by the set of places of a PN. Considering the multi-level modelling methodology described previously, this property of PNs may allow for building models for complex tasks with the risk of state explosion. If behaviours can be abstracted into places, then concurrent execution of behaviours is easily represented with PNs. Also by using PNs to model our systems we can take advantage of its structures being common to those of Generalised Stochastic Petri Nets (GSPN). If some properties are verified, the marking process of GSPN models is equivalent to a continuous time Markov chain. Thus we can, for instance, perform steady state analysis on it, as well as other types of analysis.

1.3 Outline

In chapter 2 of this report, we discuss our detailed stochastic simulator of Th cell collective, and the immunology experiments conducted to calibrate the parameters of the system. In addition, we highlight an important difference between cells and robots, that may hinder the efforts of immunology-inspired roboticists, and describe potential solutions to the problem. Chapter 3 describes a social dilemma scenario which would allow us to explore institutions in designing collective robotic systems. In addition, experiments are presented to better understand the use of institutional devices and its effect on the sustainability of a group of robots. Furthermore, in chapter 4 we present our formalism of institutions in the institutional robotics framework. We apply this formalism to a case study and build of model for the considered distributed robotic system using the robotic controller obtained from the institutions as a starting point.

Chapter 2

Multi-cellular systems and inspiration for roboticists (Task 4 and 5)

2.1 Introduction

The cells of the adaptive immune system (namely T helper cell lineage) has been extremely successful during the course of evolution as evidenced by its presence in all jawed vertebrate species. Central to their success is the important role they play in establishing and maximising the capabilities of the immune system. The Th cells as a population is capable of dynamically regulating and differentiating themselves into different sub-lineages (e.g. Th1, Th2, Th17, and Treg cells) (Harrington et al., 2005; Hori et al., 2003; Park et al., 2005; Mosmann and Coffman, 1989) to initiate the appropriate immune response. Importantly, the differentiation of naive Th cells into the different functionally distinct sub-types is decentralised, based solely on the state of the cell and local information available to it.

Progress in modern systems biology has resulted in considerable research on the genetic regulatory networks (GRN) controlling individual T helper cells (Naldi et al., 2010; Garg et al., 2009; Mendoza, 2006; Garg et al., 2007). However, an important consideration of most of these models is that an individual Th cell is analysed independent of its environment. Consequently, interesting properties of the cell population may not be characterised with such models. To this end, we developed a detailed stochastic simulator of the Th cell collective, simulating the molecular reactions of each Th cell (guided by their GRN), interactions between cells, and population growth (Tarapore and Carneiro, 2010).

The detailed nature of our simulator, coupled with the striking resemblance between cells and robots, allows the simulator to be used as a test bed for theoretical models of collective systems. Consequently, not only can the models be greatly improved to better design robot collectives, but they may also help in enriching the theory of immune response. In this chapter, we outline a classical immunology experiment simulated to calibrate the parameters of our stochastic simulator (Section 2.2). We further highlight an important different between cells and robots, that may hinder the efforts of immunology-inspired roboticists, and describe potential solutions to the problem (Section 2.3).

2.2 Stochastic simulation of Th cell collective

2.2.1 Simulation environment

A stochastic simulator of discrete events of multiple scales was developed under the scope of Task 1 of this project (Tarapore and Carneiro, 2010). Diverse repertoire of events were simulated namely, (i) molecular reaction events within a cell (Naldi et al., 2010), (ii) events leading to changes in population size, and (iii) juxtacrine and paracrine interactions between cells. The discrete event simulation of such a large and dynamically varying set of events was computationally expensive. Consequently, the priority queue at the heart of the our simulation was optimised with an implementation of a Binomial Heap data structure (Appendix A).

2.2.2 Antigen dose dependent Th cell differentiation

Our detailed stochastic simulation of the Th cell collective involves a large number of parameters (Appendix B). The calibration of the parameter values was performed by repeating certain classical immunology experiments.

The dosage of a foreign antigen can elicit a cell-mediated or humoral type of immune response. In this section, we describe an experiment that was simulated to compare with the empirical data, the number of Th cells belonging to different differentiation lineages (Th1 and Th2), under treatments with varying antigen doses (Hosken et al., 1995). The experiment demonstrated that the antigen dose used in primary cultures could directly influence Th phenotype development from naive Th cells (Th0 lineage), stimulated with dendritic cells or activated B cells as the antigen presenting cells.

Experiment protocol

An the start of the simulation, no external cytokines were added to the medium. The Th cells were activated in separate treatments with varying does of antigen presenting cells, with values ranging from 50 to 16000 cells, with 20 binding sites associated with each cell. In each treatment, the seed population of Th cells consisted of 700 Th0 cells, and 300 Th2 cells.

The population was simulated for a total period of 6 days. At the end of day 3, new antigen presenting cells were added to the population, equal to the number of APCs at the start of the simulation. This was particularly necessary in treatments with low APC doses. Following the completion of the simulation, the total number Th cells, and their lineages were recorded.

To compare the number of cells of each lineage (Th0, Th1 and Th2), we averaged for each treatment, the number of cells over 10 independent replicates, at the end of day 6 of the simulation.

2.2.3 Results

After 6 days of stimulating the Th cells with Antigen presenting cells, there was considerable variation in the number of cells recovered among the 10 treatments. At the lowest APC dosage (20), 1732 cells were recovered. By contrast, at the highest dosage (16000), the population consisted of 10943 cells. In addition, the level of paracrine cytokine IL-2, also varied amongst treatments. At the end of 24 hours, 87.3 units was found at doses of 20 APCs. In populations consisting of 16000 APCs, the level of IL-2 in the medium was much higher at 280.3 units.

The number of Th cells of different lineages in the recovered populations was influenced by the APC dosage (see Fig. 2.1). A predominance of Th2 cells was observed at low doses of APCs (50 to 500). Medium doses of APCs (1000 to 2000) resulted in a higher density of Th1 cells in the population. However, a further increase in APC dosage (4000 to 16000) resulted in a higher proportion of Th2 cells again, and a decrease in Th1 cells. Furthermore, increase in APC dosage resulted in a decrement in the number of undifferentiated Th0 cells recovered

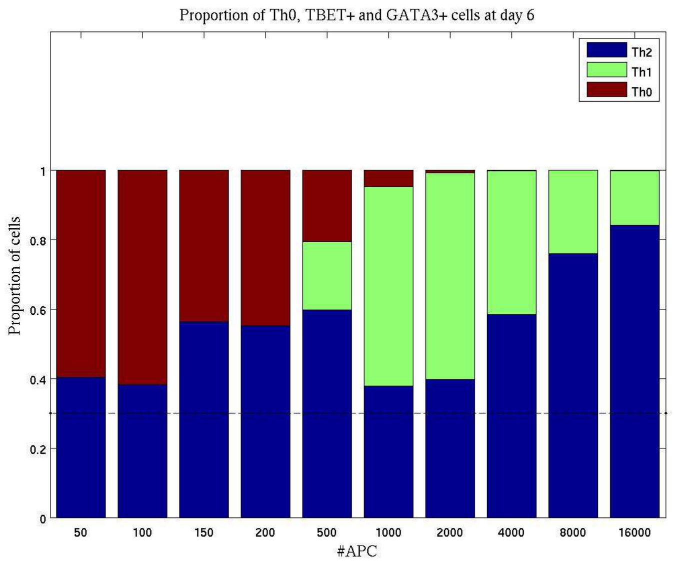


Figure 2.1: Mean number of Th0 (red), Th1 (green) and Th2 (blue) cells at the end of day 6, for different doses of Antigen presenting cells. In all treatments, the seed population consisted of 30% Th2 cells (dotted line).

(Fig. 2.1).

2.2.4 Summary and outlook

In our simulations, we investigated how variation in antigen dose (due to variation in the number of antigen presenting cells) influenced Th cell phenotype development. Our simulations revealed the predominant presence of Th2 cells with low and high antigen doses. By contrast, at medium doses, a high concentration of Th1 cells was observed in the population. These results are qualitatively similar to empirical data (Hosken et al., 1995).

In addition to experiment described above, we are simulating other classical immunology experiments (Murphy et al., 1996) to better calibrate the system parameters. We are also in the process of better understanding the dynamics of immune response while perturbing the system, such as rewiring the GRN inside some/all cells, changing the parameters of the cell-to-cell interactions, and adding or killing cells.

The detailed nature of our simulations of Th cell collectives, allows them to bear a striking resemblance to a robot collective. For example, communication is simulated on contact (cell conjugation), at medium range (juxtacrine cytokine) and at long range (paracrine cytokine). Consequently, our simulator can be used as a test bed to evaluate detailed mathematical models of a collective. This would serve a dual purpose, (i) an improved theory of immune response, and (ii) allow roboticists to evaluate, test and improve these mathematical models, which can then be utilised to design better robot collectives. However, an important difference between cells and robots is that cells can proliferate, while robots can not. In the next section, we address this issue and suggest a possible replacement for cell proliferation, while retaining the decentralised and self-organising nature of the adaptive immune system.

2.3 Transitioning from cells to robots

The adaptive immune system is characterised by its ability to differentiate between self and non-self antigens. The system is able to rapidly deploy a response to antigens that are foreign to the host, while continually tolerating antigens that are part of the self. Roboticists would like to emulate this property in their collective. Cell proliferation (division of a single cell into two daughter cells) is the fundamental mechanisms used by the adaptive immune system to orchestrate these responses. However, a fundamental difference between cells and robots is that robots cannot proliferate (except some modular robotic systems in which composite entities may replicate when sufficient individual units are available (Zykov et al., 2005)). In this study, we therefore, evaluate if and when *recruitment* can be used to emulate the process of cell proliferation. Proliferation is simulated by the transfer of a controller from a "proliferating" robot to a "free" one. In order to determine if and when recruitment can be used to emulate proliferation with reasonably accuracy, we compare two system: one that relies on proliferation and one that relies on recruitment,respectively. We model both systems analytically and we implement them in a stochastic simulation environment.

The mechanisms underlying the tolerant and immune response of the adaptive immune system has been explained by the Crossregulation Model (CRM) (Leon et al., 2003). An ODE-system of the population dynamics (using proliferation) has demonstrated very clearly the properties of this model. We would like to see if recruitment instead of proliferation can replicate these properties. In Section 2.3.1, we discuss the Crossregulation model and its implementation in our study utilising both proliferation and recruitment. Section 2.3.2 explains our stochastic simulation environment. Section 2.3.4 presents the resulting cell population dynamics with proliferation and recruitment. Finally, in Section 2.3.5, we go on to discuss the main issues concerning the application of adaptive immune system-inspired approaches to realistic multirobot systems and how we may overcome these issues.

2.3.1 Crossregulation model

The Crossregulation model describes the population dynamics of Th cells in the periphery, taking into account three cell types.

1. Antigen presenting cells (A) that display a processed antigen on their sur-

face.

2. Effector cells (E) that can potentially induce an immune responses to foreign pathogens, or an autoimmune response to self-antigens.

3. Regulatory cells (R) that suppress effector cells preventing their clonal expansion.

In our implementation of the CRM, effector and regulatory cells can be in one of three states, namely free (not bound to an APC), conjugated to an APC, or activated. Effector cells in the free, conjugated and activated states are denoted by Ef, Ec and Ea respectively. Similarly, regulatory cells in the free, conjugated and activated states are denoted by Rf, Rc and Ra respectively.

The interactions between the effector and regulatory T cells, with the Antigen presenting cells is described with the following system of ODEs. Total number of conjugated sites in the population is C is given $C = Ec + Rc$, where Ec and Rc denote the number of conjugated effector and regulatory cells. Changes in the conjugated effector and regulatory cells is given by the following equations.

$$Ec' = kbAEf\left(1 - \left(\frac{C}{As}\right)^s\right) - kubEc - \delta_e Ec \tag{2.1}$$

$$Rc' = kbARf\left(1 - \left(\frac{C}{As}\right)^s\right) - kubRc - \delta_r Rc \tag{2.2}$$

In the above two equations, A is the number of APCs, s is the number of binding sites per APC and, E_f and R_f represent the density of free effector and regulatory cells respectively. The free cells bind to A at rate kb and unbind at rate kub . In addition, δ_e and δ_r indicates the death rate of effector and regulatory cells respectively.

According to the CRM, conjugated effector cells are able to proliferate only when they do not have a neighbouring regulatory cell bound to the same APC. If we denote by P_e, the probability that a conjugated effector cell has no neighbouring regulatory cell, the equations describing the change in free and activated effector cells is;

$$Ef' = -kbAEf\left(1 - \left(\frac{C}{As}\right)^s\right) - \delta_e Ef + kub(1 - P_e)Ec + 2\sigma_e Ea \tag{2.3}$$

$$Ea' = kubP_eEc - \sigma_eEa - \delta_eEa \tag{2.4}$$

In the above two equations, conjugated effector cells with no neighbouring regulatory cell (P_eEc) are selected for activation. These activated effector cells proliferate at rate σ_e. By contrast, conjugated effector cells with a neighbouring regulatory cell $(1 - P_e)Ec$ unbind without proliferation.

Conjugated regulatory cells on the other hand proliferate only upon being colocalized with an effector cell. If we denote by P_r, the probability that a conjugated regulatory cell has a neighbouring conjugated effector cells, the equations describing the change in free and activated regulatory cells is;

$$Rf' = -kbARf\left(1 - \left(\frac{C}{As}\right)^s\right) - \delta_rRf + kub(1 - P_r)Rc + 2\sigma_rRa \tag{2.5}$$

$$Ra' = kubP_rRc - \sigma_rRa - \delta_rRa \tag{2.6}$$

In the above two equations, conjugated regulatory cells with a neighbouring effector cell (P_rRc), are selected for activation. These activated regulatory cells proliferate at rate σ_r. However, conjugated regulatory cells with no neighbouring effector cell $(1 - P_r)Rc$, unbind without proliferation.

In our system of ODEs, the probabilities P_e and P_r are derived following a multinomial approximation (Evans et al., 2000). This approximation is valid as long as the total number of binding sites As is much larger than the number of sites per APC. The probabilities, P_e and P_r can be given by the following two equations, for A antigen presenting cells, with 3 binding sites per cell.

$$P_e = \frac{(Rc - 3A)^2}{9A^2} \tag{2.7}$$

$$P_r = \frac{(6A - Ec)Ec}{9A^2} \tag{2.8}$$

The table 2.1 indicates the parameters with description, used in the analytical model, and their values.

Cross-regulation model with recruitment

Cell proliferation is a fundamental mechanisms used by the adaptive immune system. Since robots cannot proliferate, we have designed and studied a model of

Table 2.1: Parameters for differential equation system for the Crossregulation model.

Parameters	Description	Value (a.u.)
kb	Conjugation rate of T cells to Antigen presenting cells	10^{-6}
kub	Dissociation rate of T cells from Antigen presenting cells	10^{-3}
σ_e	Proliferation rate of effector cells	10^{-4}
σ_r	Proliferation rate of regulatory cells	10^{-4}
δ_e	Death rate of effector cells (proliferation model), or transitioning rate of effector cells to idle type (recruitment model)	10^{-5}
δ_r	Death rate of regulatory cells (proliferation model), or transitioning rate of regulatory cells to idle type (recruitment model)	10^{-5}
kr	Recruitment rate of idle cells	10^{-4}

cross-regulation in populations with a fixed number of agents that use *recruitment* instead of proliferation.

In our recruitment model, in addition to effector cells of type Ef, Ec, Ea and regulatory cells of type Rf, Rc, Ra, we define a prefixed total number of cells N. Cells that are neither effector or regulatory are considered as idle (I), where

$$I = N - Ef - Ec - Ea - Rf - Rc - Ra \tag{2.9}$$

The system of equations for binding and unbinding between effector and regulatory cells is the same as in the proliferation model. However, cell proliferation is now replaced by the recruitment of an idle agent into the proliferating cell's type, at rate kr (see Table. 2.1). Similarly, cell death is simulated by its transition to an idle state. Consequently, the equations describing the changes in free and activated effector cells is;

$$Ef' = -kbAEf\left(1 - \left(\frac{C}{As}\right)^s\right) - \delta_e Ef + kub(1-P_e)Ec + 2\sigma_e krEaI \tag{2.10}$$

$$Ea' = kubP_eEc - \sigma_e krEaI - \delta_e Ea \tag{2.11}$$

Similarly, the equations describing the changes in free and activated regulatory cells is;

$$Rf' = -kbARf\left(1 - \left(\frac{C}{As}\right)^s\right) - \delta_r Rf + kub(1-P_r)Rc + 2\sigma_r krRaI \tag{2.12}$$

$$Ra' = kubP_rRc - \sigma_r krRaI - \delta_r Ra \tag{2.13}$$

2.3.2 Simulation environment

We use a stochastic, spatial, discrete-time simulation environment (Tarapore and Christensen, 2010) in which all entities are point-sized. Four types of entities are simulated, namely effector agents, regulator agents, idle agents, and antigen presenting cells (APCs). The three types of agents move at a speed of 0.3 units/cycle and change direction with a probability of 0.01 each cycle. APCs do not move.

When a floating agents detects an APC within range and with a free binding site, it conjugates to the APC. The agent remains close the static APC while conjugated. When an agent detaches from an APC, it recommences random walk. In the recruitment model, an activated agent will simulate proliferation by recruiting an idle agent. The recruited agent (the daughter) will become an agent of the same type as the agent that recruited it, that is, either an effector agent or a regulator agent.

Agents have a maximum perception range of 1 for APCs and 6 for other agents. The other parameters used for our the stochastic simulation of the recruitment-based cross-regulation model can be seen in Table 2.2.

2.3.3 Statistical analysis and numerical integration

At the start of the simulation, the population consisted of 60 effector cells ($E_f = 60$) and 40 regulatory cells ($R_f = 40$). In separate experiments, the number of

Table 2.2: Parameters for stochastic simulation of the Crossregulation model.

Parameters	Description	Value (a.u.)
v	Velocity of cell	0.5
r_A	Detection radius for Antigen presenting cell	1
L	Area of experiment arena	10^6
kub	Dissociation rate of T cells from Antigen presenting cells	10^{-3}
σ_e	Proliferation rate of effector cells	10^{-4}
σ_r	Proliferation rate of regulatory cells	10^{-4}
δ_e	Death rate of effector cells (proliferation model), or transitioning rate of effector cells to idle type (recruitment model)	10^{-5}
δ_r	Death rate of regulatory cells (proliferation model), or transitioning rate of regulatory cells to idle type (recruitment model)	10^{-5}
r_I	Detection radius for idle cell	6

Antigen Presenting Cells (A) was set at 20, 40, 60, 80 and 100. Each APC had 3 binding sites associated with it. Populations were simulated for $4x10^7$ time-steps at which point the number of effector and regulatory cells had converged in all experiment conditions. In the case of the recruitment model, an identical set of experiments was conducted, with the total number of cells N fixed at 20100.

Since the population dynamics appeared similar at high levels of APCs, transitions in cell population has been show for the extreme cases of 20 and 100 APCs. To compare the number of effector and regulatory cells ($E = Ef + Ec + Ea$) and ($R = Rf + Rc + Ra$), we illustrate for each treatment, the number of cells for each of the 10 independent replicates.

In order to compare the results of the stochastic simulation with the analytical model, the numerical integration of the set of ordinary differential equations was performed using the Explicit Runge-Kutta method (Butcher, 2003). The procedures used to compute the numerical integration, and the stable states illustrated in the Bifurcation Diagram, were provided by the Mathematica software framework (Wolfram Research, 2008).

2.3.4 Results

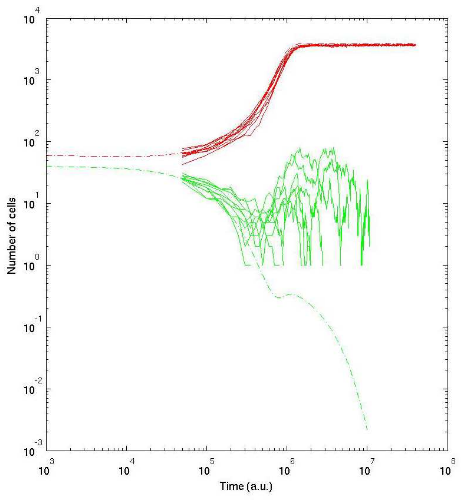
The results of the stochastic simulation indicate the population of effector and regulatory cells to be in one of two stable states (Fig. 2.2), (i) an immune state, characterised by the presence of only effector cells, and (ii) a tolerant state, consisting of a dominant population of regulatory cells and few effector cells. In both models of proliferation and recruitment, the system converge to an immune state in the presence of few APCs (20 cells in Fig. 2.2b and d). In addition, an increase in the number of APCs (100 cells in Fig. 2.2b and d) present in the environment resulted in the system converging to a tolerant state

Our results also indicate that the suggested analytical model can explain the population dynamics of the stochastic simulation for both proliferation and recruitment (Fig. 2.2). In the steady state conditions, the bifurcation diagrams of both these models appear qualitatively similar. At low APC doses, the system could only reach an immune state. Additionally, upon increasing the APC dosage, the system could jump to one of two stable states (immune or tolerant), depending on the configuration of the seed population, with the presence of an unstable state in between. However, the bi-stability while seen in ODE model, has yet to be verified with the stochastic simulation.

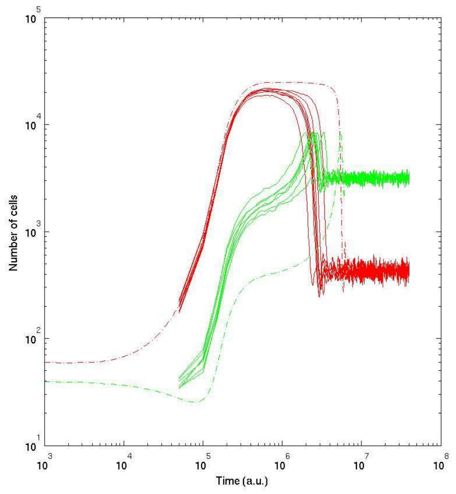
2.3.5 Summary and outlook

In this section, we highlighted the problems with utilising immune system inspired algorithms for robot collectives. We further propose an alternative to proliferation, namely recruitment, that may be used with a fixed number of agents/robots. Our stochastic simulations comparing proliferation and recruitment were demonstrated for the Crossregulation model (Leon et al., 2003), and indicated qualitatively similar characteristics. In addition, the corresponding analytical models also demonstrate the properties of the Crossregulation model and are able to well explain the population dynamics of the stochastic simulations.

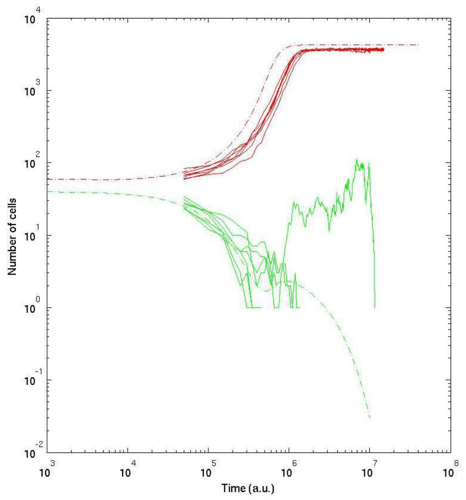
An important issue to note when utilising the recruitment model is the large number of agents needed. In our simulations, 20000 idle agents had to be pro-



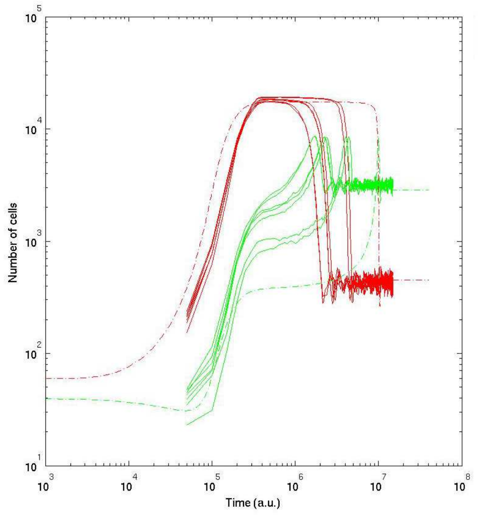
(a) Proliferation mode, 20 APCs)



(b) Proliferation mode, 100 APCs)



(c) Recruitment mode, 20 APCs



(d) Recruitment mode, 100 APCs

Figure 2.2: Number of effector (red) and regulatory (green) cells for different number of Antigen presenting cells, with the stochastic simulation (solid), and analytical model (dashed).

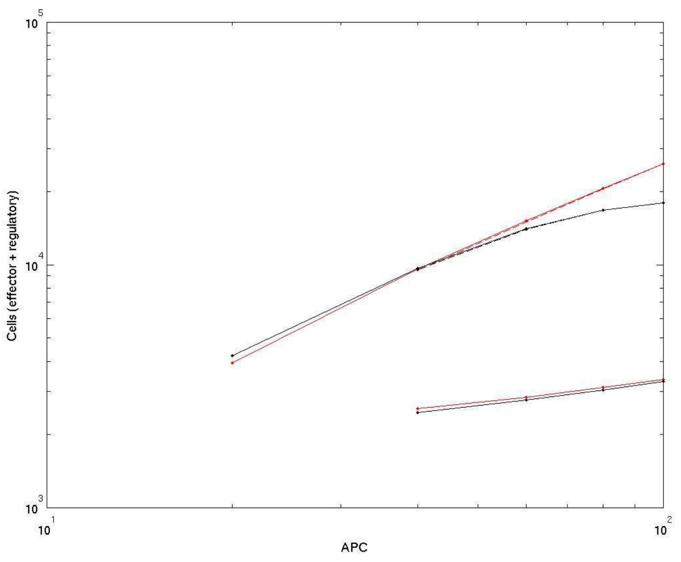


Figure 2.3: Total number of cells in stead state conditions for different doses of Antigen presenting cells, with the proliferation (red) and recruitment (blue) models. Stable and unstable states are indicated by solid and dashed lines respectively.

vided to the population. Addition, an increase in Antigen presenting cells would demand an even higher number of these agents. Our current effort are involved in reducing the number of idle agents needed, while retaining the characteristics of the Crossregulation model.

The adaptive immune system may be considered to allow tolerance to emerge from the local interactions between specific cell types, and largely independent of specific characteristics of antigen. This characteristic has been explained by the Crossregulation model, involving local interactions between effector, and regulatory cells at Antigen presenting cells. In the next step, we would like to emulate the CRM in a novelty detection scenario, wherein the robots as a collective may be able to discriminate between stimuli that are part of their environment (self-antigens), and that which is newly introduced (foreign-antigens). The antigens formulated, can be considered as a sequence, encoding important sensory information. In such a scenario, antigen presenting cells may be considered as agents that process and present the antigen sequence to effector and regulatory agents. We hope that such a system may be able to discriminate between self and non-self aspects of its environment in a robust manner, and without the need for any specific information on the antigens.

2.4 Mathematical models

The differential equation model presented in this chapter explains populations dynamics resulting from interactions between three cell types (antigen presenting, effector and regulatory cells). Based on the nature of an individual cell's "controller" at the micro-level, we were able to come up with a top-down, macro-level model, to analyse the dynamics of the system. However, our existing mathematical model may not scale up to more complex systems involving a higher number of micro-level states, the possibility of adding new states, and scenarios with more complex (heterogeneous) environments. In such cases, as (i) detailed multi-cellular stochastic simulations, or (ii) realistic collective robotic scenarios, e.g., the novelty detection case discussed earlier, another mathematical formalism is needed. In the fourth chapter, we present a stochastic petri net formalism (a bottom-up approach) that may be adequate to scale up with more complex simulations.

Chapter 3

Institutional inspired robotics (Task 5)

The institutional inspired approach to collective robotics tackles the problem of system design differently from a bio-inspired approach. Agents in institutional robotics (IR) are comparatively more deliberative, their behaviour being harder to capture. An IR approach allows us to consider agents as more complex then reactive agents. In this approach, agents can actually modify the rules that guide their behaviour, via cooperative decision-making processes. Coordination between agents is achieved by this self-regulation of social interactions since the robots know not only how to behave in a given scenario but also what to expect from other robots and the environment.

In this chapter we describe our work towards using institutions in a collective robotics systems. Section 1 details a social dilemma scenario to experiment with institutional concepts. An exploration of reinforcement learning in the decision making process of institutional agent is then presented in section 2. In section 3 we present our implementation of a decentralised and institutional decision making devices in a collective robotics task scenario. Finally, section 4 highlights our conclusions and outlines the experiment scenarios to be investigated in the future.

3.1 Social dilemma scenario

A "social dilemma" exists where there are no prima facie way to make actions guided by the pursuing of (perceived) individual utility easily compatible to (perceived) collective utility. A simplified model of common-pool resources could be the basic setup (see Fig. 3.1), because it allows taking into account two important features of most complex social situations at human level: the problematic sustainability of the resources and the temptation to free-ride.

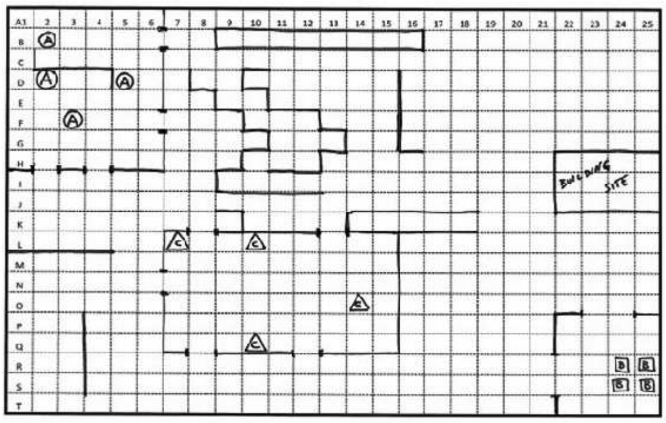


Figure 3.1: An example scenario demonstrating a collective task. The environment consists of objects of type A, B and C scattered randomly. Agents have to forage for these objects and bring them to the building site.

The basic elements of such a scenario are the following:

1. the basic task is to construct as many specimens of a "virtual object" as possible by assembling, in a specified way, tokens of different resources that can be found in the environment (components A, B, and C to be assembled as virtual objects $A+B+C$); the variation of this basic task should be easily implementable;

2. the experiment takes place in a 2D space; within this virtual environment, there is the "building site" (where the assembling takes place), and fields of sources of the different components needed to build the virtual object;

3. individual robots try to maximise private utility functions (delivering components to the building site); the system has a collective utility function, mainly directed to the global task of building as many virtual objects as possible; the basic social dilemma springs from these two kinds of utility functions;

4. the resources system is a renewable resource system; it has a specific replenishment rate; the rate of withdrawal must be balanced with the replenishment rate to avoid (reversible) damage or (irreversible) destruction of the resource system; initially, robots don't have information neither about the localisation of the spots where components can be collected nor about replenishment rates of components' sources.

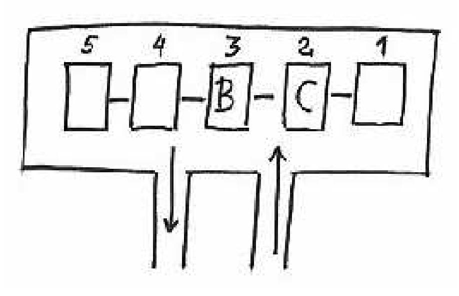


Figure 3.2: An example of the social dilemma occurring at the assembly site. The robots are required to assemble virtual objects $A + B + C$. A conflict occurs between delivering A to slot 4 (higher collective reward), in contrast to delivering to slot 5 (higher individual reward).

The conflict between individual and collective utility occurs at the building site (see example Fig. 3.2). Let us consider that delivering a component to slot 5 rewards 5 times more than delivering the same component to slot 1. In this example situation, an A component should be delivered to slot 4 to immediate completion of an object, but delivering that component to slot 5 is more rewarding to the individual agent carrying it.

In order to experiment with institutional roles/positions as "packing information" and "packing decision" devices, we focus on the functioning of the building site and introduce an "assembler" there. The assembler works in the antechamber of the building site as follows. It accepts from any three robots the delivery of three components (A, B, C) allowing immediate construction of an object. It calculates the most profitable way to put these components to the slots, giving priority to collective utility. Consequently, the assembler calculates the total reward corresponding to this delivery, takes to itself a certain percentage, and divides the remainder equally by the contributing robots. After a while, the system calculates the number of pieces completed during a given period. Another run of the experiment goes without the assembler and letting each individual robot pursuing its individual utility. In such a decentralised scenario, individuals decide for themselves the assembly slot to place the object. The number of pieces completed with both the ***decentralised***, and the ***institutional*** regimes are compared. We would like to experiment if (and at what extent) roles/positions can improve both individual and collective utility and parameters influencing their performance (for example, the percentage retained by the assembler may vary, and it may impact the result: a too expensive assembler can make the device fail when compared to purely individual behaviour).

3.2 Reinforcement learning in institutional robotics

The influence of the short and long term rewards on the agents decision is dependent on the discount factor (γ), that is individual to each agent. This discount factor represents the time horizon at which the agent perceives its reward. An agent with a lower discount factor would prefer short term reward. By contrast, a high discount factor agent would make its decisions in order to obtain a long term reward instead. The relationship between discount factor an agent behaviour bears a strong relationship to reinforcement learning (RL) and decision making theory. With the usage of RL, the agents behaviour does not need to be hardcoded, avoiding an artifacts that may be introduced by the experimenter. In this section, we investigate how reinforcement learning may be introduced into our social dilemma scenario.

The main goal of the experiments to study the capacity of an institutional device to transform an unsustainable system into a sustainable one, where the threat of unsustainability is due to the behaviour of short-sighted agents, without modifying the inner world of the individuals. To that effect, we need first to study the impact of the percentage of short-sighted agents in a population of short-sighted (i.e., with small discount factors γ) and far-sighted agents (i.e., with discount factors γ 1), in the expected accumulated reward (EAR) of the task at hand. In our collective task, the expected accumulated reward concerns the number of pieces completed in a given time horizon, e.g., T. The experiments are be carried out for the decentralised and institutional regimes. We would like to investigate if the assembler intervention turns an unsustainable system into a sustainable one. It is important to note that while in the decentralised case the agents use RL to determine their optimal policy, in the institutional case the policy is pre-determined and applied by the assembler. However, in both cases the agents receive the rewards of every action taken of depositing parts in the bin slots.

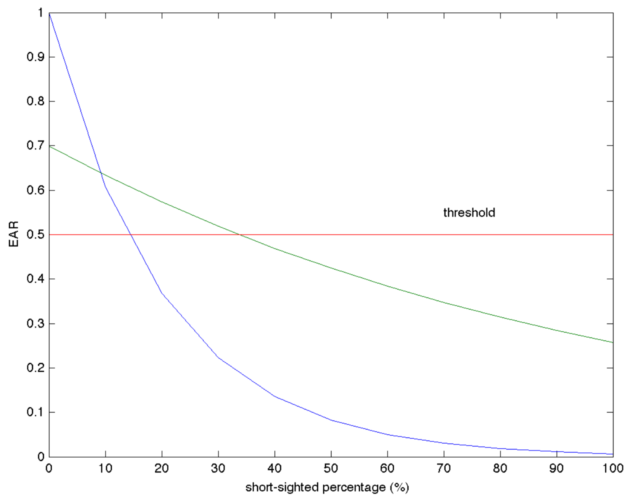


Figure 3.3: Institutional (green) vs Decentralised (blue) typical expected performance, concerning the number of pieces produced in a time interval T with and without an assembler, respectively. For a $EAR = 0.5$ threshold, the percentage of short-sighted agents above which the performance degrades below the threshold is 18% in the Decentralised case, but increases to 35% in the Institutional case. Note that in the interval $[0,\ 10\%]$, the number of pieces produced within the time interval T is larger without a mediator.

We hypothesise that the EAR over the time horizon T will decrease in both cases with the percentage of short-sighted agents in the population, because the larger the number of greedy agents, the lower the number of complete pieces completed in the same time interval. In the Institutional scenario, despite the fact that the assembler performs the actions, the agents will still receive rewards and the same situation will probably occur. However, we claim that the impact of the percentage of short-sighted agents will be less, meaning that the EAR will decay at a slower rate than for the Decentralised case when the percentage of short-sighted agents increases. This results from the fact that the mediator/assembler is by itself speeding up the piece assembly process. So a given performance (EAR) threshold is met by a population with a larger number of short-sighted agents in the Institutional case (see Fig. 3.3).

It is interesting to note that the rate of decay of the EAR plot in the Institutional case may be initially faster than the rate of decay of the EAR plot for the Decentralised case, and get below it for a given interval of short-sighted agents densities (see Fig. 3.3). This would be intuitively explained by the fact that, for a low number of agents, the population may not need a coordination artifact (i.e., the mediator/assembler).

An upper level of learning may be introduced in a second experiment, so as to let the population be running Decentralised and Institutional regimes concurrently (or in a time-sharing manner, running one for some time interval, then the other for another time interval and so on and so forth) and decide over time (for a given percentage of short-sighted agents) whether to use Decentralised or Institutional mechanism. The decision to use an Institution would be interpreted as the collective deciding a (pre-built) institution to improve its performance.

3.3 Implementation of decentralised and institutional approaches

As a first step towards exploring the effects of the decentralised and institutional regimes on the sustainability of the population, we consider that the behaviour of the agents is hardcoded. Additionally, the institutional device is limited to processing information at the assembly site. However for future investigations, we could imagine the integration of RL, and the need for other institutional devices if the resources in the environment are limited with a specific replenishment rate, and the rate of withdrawal must be balanced with the replenishment rate to avoid (reversible) damage or (irreversible) destruction of the system. In this implementation, we focus on an institutional device for decisions made at the assembly site.

3.3.1 Defining the system

A population of agents has to produce pieces $(1-2-3 \ldots n)$, composed of individual components of type $1, 2, 3 \ldots n$. These n items are scattered across a building site, and have to be brought to a pre-designated assembly site for the construction of the final object.

The assembly site has m slots marked $1, 2, 3 \dots m$, and equal to the short-term reward the agent receives for placing the item in the slot. On returning to the assembly site, the agent may instead choose to place its load so that a sequence $(1 - 2 - 3 \dots n)$ can be completed. Such robots may not receive a high short-term reward, but may receive a long-term reward upon the completion of a sequence. The influence of short and long term rewards is dependent on the discount factor (γ), that is individual to each agent. Sustainability of the population is defined by the number of assembled objects it can produce.

Assembly site

In our experiments, the state of the assembly site S can be defined by the following $m + 1$-tuple.

$S = (C_m, C_{m-1} \dots C_1, F)$

where m is the number of slots at the assembly site, C_i indicates the component in each slot $i \in \{1 \dots m\}$, and F is the type of component brought by an agent to the assembly site.

$C_i \in \{1, 2 \dots n, \phi\}$, where n is the number of different types of components at the building site. The slot i of the assembly site can hold a component of type $1, 2 \dots n$, or it may even be empty (i.e., $C_i = \phi$). The component brought by an agent to the assembly site $F \in \{1, 2 \dots n\}$.

The completion of a piece requires the placement of components in consecutive slots of the building site, so as to get the exact sequence of components $(1, 2 \dots n)$.

Agent

The population to be evaluated consists of T agents. An individual agent A_i is defined as follows.

$A_i = (\gamma_i, F_i, AR_i)$ for $i \in \{1, 2 \dots T\}$,

where γ_i is the discount factor, F_i is the type of object collected by the agent, and AR_i is the agents accumulated reward.

3.3.2 Implementation of decentralised decision making

In a decentralised decision making scenario (Appendix C: algorithm 1), agents decide on their own where to place the foraged component. For each of the slots j, the agent computes the approximate utility u_j it may receive if the component is placed in slot j (Appendix C: algorithm 2).
$u_j = RI_j + P_j(RC \times \gamma)$

The utility is the sum of the immediate reward associated with the slot ($RI_j = j$), and the discounted collective reward ($RC \times \gamma$) when the component in the slot allows the construction of a piece sometime in the future (i.e. $P_j = 1$ else $P_j = 0$). P_j (see Appendix C: algorithm 3) can be considered as the prospect of completing a piece when the component is placed at slot j. Finally, the agent chooses the available slot that maximises its utility. Fig. 3.4 illustrates the flow of components from the building site to the assembly site, controlled by the policy of individual agents.

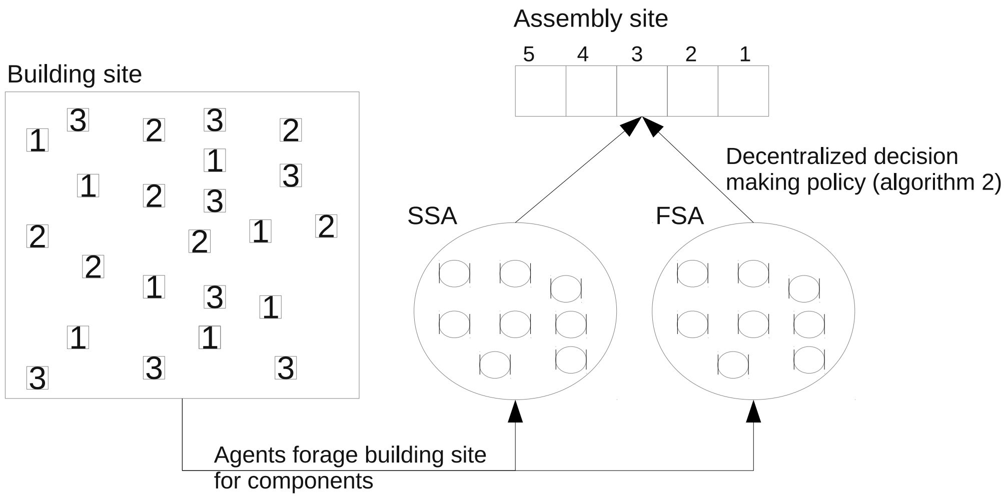

Assembly site
5 4 3 2 1
Building site
Decentralized decision making policy (algorithm 2)
SSA
FSA
Agents forage building site for components

Figure 3.4: Decentralised decision making scenario with number of slots $m = 5$ and number of component types $n = 3$. Shortsighted (SSA) and farsighted agents (FSA) forage the building site for components. Foraged components are consequently placed in the assembly site in slots determined by a decentralised decision making policy.

Setting parameters of the experiment: The discount factors for short-sighted (γ_{SSA}) and far-sighted γ_{FSA} agents is fixed at 0.1 and 0.9. In addition, preliminary experiments will be conducted to tune the collective reward RC. The reward value for piece completion is be set to ensure that far-sighted agents will be compensated for taking into account the long-term future reward of completing a piece, compared to the short-sighted agents.

Experiments to perform: We would like to conduct independent experiments while varying the proportion of short sighted agents in the population, and the complexity at the assembly site. The complexity can be varied with the number of slots at the assembly site m, and the number of component types n.

The experiments would allow us to investigate the density of short-sighted agents (SSA) in the population that would drop the number of completed objects below a threshold and require the need for institutions. Furthermore, the influence of the complexity of the assembly site on this density of SSA will also be investigated.

3.3.3 Implementation of institution based decision making

In this previous case study we have selected a suitable collective reward (RC), and seen the need for sustainability in scenarios with a high density of short sighted agents. We now bring institutions into the picture . In the institutional scenario (see Appendix C: Algorithm 4), the decision of where to place the component is made by the assembler. The simulation proceeds as follows. Individual agents gather components from the building site and proceed to the assembler. If the assembler requires the component presented by the agent, it keeps it and the agent is available to bring more components to the assembler. However, if the assembler has no need for the component, the agent is made to wait at the assembler until its component is needed for piece completion. Consequently, the total number of agents T in the population is now divided into agents waiting at the assembler W, and available agents A.

When the assembler has all the components necessary for completion of the piece, it places the components $1, 2 \ldots n$ in slots $m, m-1 \ldots m-n+1$ (see Appendix C: Algorithm 5). After taking a assembler fee AF, the remaining collective reward $(RC - AF)$ is distributed amongst the contributing agents. The contributing agents also receive an individual 'immediate' reward corresponding to the assembler slot index at which their component was placed. Fig. 3.5 illustrates the flow of components from the building site to the assembly site, controlled by the assembler. The assembler also controls the agents that are available for foraging and those that are held waiting in a queue.

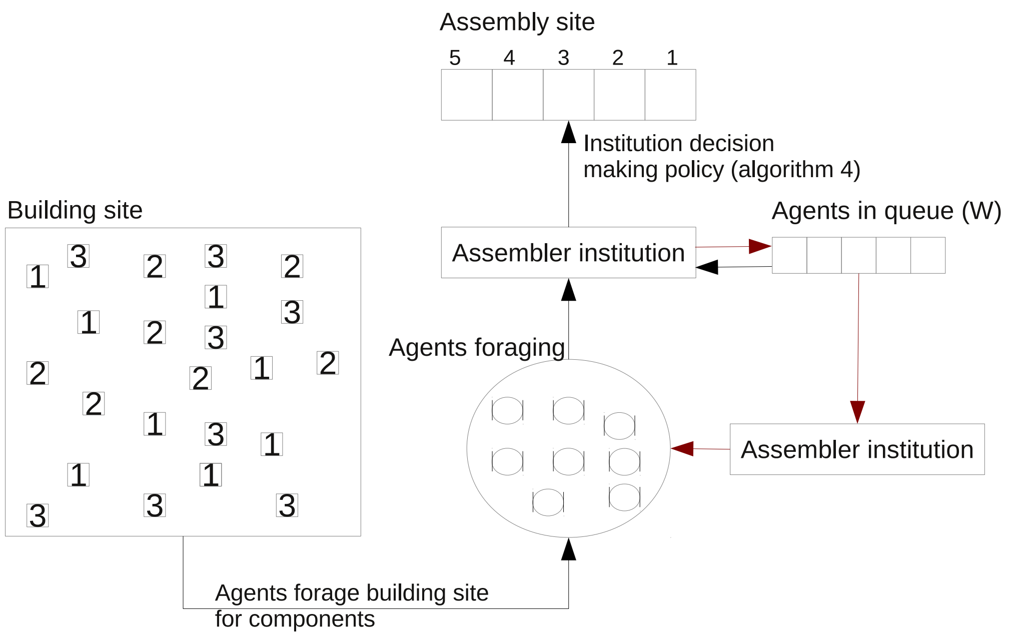


Figure 3.5: The flow of components (black arrows) and agents (red arrows) in an institution decision making scenario. Available agents forage the building site for components. The foraged component can be accepted by the assembler and the agent then resumes foraging. However, if the component is not needed, the agent in placed in a queue until the component is required by the assembler. Assembler places the component in slots determined by a institution decision making policy.

Experiments to perform: We would like to conduct independent experiments while varying the assembler fee (AF), and the complexity at the assembly site. These experiments would allow us to investigate if an institution can transform an unsustainable system into a sustainable one. In addition, how the complexity at the assembly site would influence this transformation.

In the current case study, we do not expect the assembler fee (AF) to influence the number of pieces completed. Instead, the fee would come into play when agents can collectively decide the use of an institution.

3.4 Conclusions and future work

The second task report (Silva, 2010) suggested a set of experiments with institutional concepts. In this chapter, we have presented part of that scenario i.e., the agents behaviour at the building site. The main goal of the experiments is to study the capacity of an institutional device to transform an unsustainable system into a sustainable one. In this case, a sustainable system is a system producing a level of resources (“energy”) high enough to “feed” all the agents. The production of “energy” depends on the objects assembled at the building site. To model this, we assume a threshold for the minimal production level the system must guarantee.

The threat of unsustainability is due to the behaviour of short-sighted agents (SSA). When deciding how to deliver a component to the building site, SSA are able to take into account the reward they can get immediately from delivering a component to the building site (type 1 reward), but are not able to take into account the reward they can receive later when that component is part of a new assembled object (type 2 reward). The decision-making of far-sighted (FSA) agents take into account both types of reward. SSA are agents with small discount factors; FSA are agents with discount factors close to 1.

Our study would compare two different basic versions of the scenario, namely the Decentralised and Institutional regimes. In the Decentralised case, each agent guided only by his short-sighted or far-sighted decision mechanism, delivers the components to the building site. By contrast, the Institutional case involves a mediator working at the building site that collects from other agents a right combination of components to complete an object; finishes the assembling at the building site; collects, at the same time, both type 1 and 2 rewards due to those components, retains a “service tax”, and gives the remaining to the agents who delivered the parts.

In our experiments, we do not want the system to be sustainable only by an external decision of the experimenter. Consequently, in future work, each agent compare the reward he received at both decentralised and institutional scenarios, and they all decide by majority vote which scenario they prefer. Each agent will vote for the scenario that paid him the greater reward. The system will be self-sustaining if the most productive scenario is the one preferred by the majority of individual agents. Agents do not need to be altruistic: we would like to know if an institutional device can make the system self-sustaining as a collective, without the need of external intervention âĂŞ and without changing SSA to FSA. This would indicate that we are not working at psychological individual level, but at collective, organisational level.

In this chapter we have also discussed endowing the agents with some kind of unsupervised learning. Since the discount factors make a connection with Reinforcement Learning, we considered using RL. We would like to explore this implementation in the future. However it is important to note, that it may be difficult to model all relevant aspects of the institutional background with RL. For example, there is no “objective world” within the scenario, because the same action can have different outcomes depending on future actions of other agents (more “cooperative” or more “individualistic”). In addition, it would take a long time for an agent to visit all the relevant possible states to correctly assess the situation. Another option we would like to assess in the future is to let agents compare multiple trials of both the decentralised and the institutional scenarios and then let them decide (by majority vote) based only on the sum reward each received. Such a scenario could be considered as a form of collective learning, unbiased by the experimenter and with the population deciding on the scenario most beneficial for its sustainability.

Chapter 4

Modelling institutions with Petri net formalism (Task 6)

4.1 Introduction

One of the goals of our research is to formalize the concepts of Institutional Robotics (IR) from a computer science perspective. In (Pereira et al., 2011), we presented a formalization of institutions, using Executable Petri Nets as an abstract representation. This formalism allows the design and execution of institutions in robots, so as to obtain behaviors capturing social interactions of interest. Our method produces, from a set of institutions, a robot controller able to execute a desired task. A further goal of our research is to develop models of the IR approach, in order to quantitatively and qualitatively predict the performance of the system, carry out possible optimizations, formally analyze performance bounds, and verify general system properties (e.g. liveness).

We are interested in assessing if institutional controllers can be used for modeling the distributed robotic system they control by providing the necessary model structures. We use the Petri Net structures of the institutions designed for our case study to derive a macroscopic model that captures the mean-field dynamics of the distributed robotic system. We compare our model predictions with results obtained in realistic simulation.

4.2 Institutional Robotics Models

Our intended approach to modeling in the Institutional Robotics framework consists on the employment of our formalism of institutions (Pereira et al., 2011), in order to obtain a controller that allows robots to execute the task at hand, and subsequent construction of a stochastic model based on that controller. In this section we will detail separately the formalism of institutions and the stochastic model construction. As an example for both we will use the wireless connected swarm case study.

4.2.1 Institutional Agent Controller

Starting from the concept of institutions as coordination artifacts (Tummolini and Castelfranchi, 2006) we model them using a formal representation, leading to a standard design and execution platform (in real robots, realistic simulations, and multi-agent systems). Institutions can be considered as behavioral abstractions meant to improve coordination activities. They represent the basic building blocks for creating shared coordinated working environments. Considering the three main properties of coordination artifacts mentioned in (Omicini et al., 2004), specialization, encapsulation, and inspectability, we propose to use Petri Nets (PN) (Cassandras and Lafortune, 2008) as the formal framework.

Our aim is to formalize institutions as Petri Nets both for design and execution of robotic controllers. This means that we need to take into account robot actions and sensor readings. We consider three sets of building blocks that will allow us to design our controllers.

The set Act contains all robot actions (combinations of two or more primitive actions can be considered as actions). The set Cdt contains boolean conditions that can be verified by checking sensor readings. Finally, the set Pac contains "parameter actions", which are auxiliary actions not concerning actuators but that only modify variables needed for the actions in Act.

We are now able to define our own version of Petri Nets used for execution of our robotic controllers.

Definition: An *Executable Petri Net* (EPN) is a Petri Net (P, T, A, w, X) where:

- each place $p_i \in P$ has an associated action $a_i \in Act$;

Algorithm 4.1 Execute Petri Net

```
 1: repeat
 2:    for all enabled transitions $t_i \in T$ do
 3:       if associated condition $c_i$ is true then
 4:          run associated parameter action $pa_i$
 5:          fire transition $t_i$
 6:       end if
 7:    end for
 8: until no transition has fired
 9: for all marked places $p_i \in P$ do
10:    run associated action $a_i$
11: end for
```

- each transition $t_i \in T$ has an associated condition $c_i \in Cdt$ and an associated parameter action $pa_i \in Pac$.

The basic intuition behind this definition is that by associating actions with places we are able to define which actions are to be executed at each time step. This is done simply by checking if the corresponding place is marked. By associating transitions with conditions verified by sensor readings we trigger state changes in the Petri Net due to changes in the robots environment. Algorithm 1 is performed by the robots at each time step, allowing the robots to execute the behavior designed in an EPN.

Institutions are formalized as coordination artifacts in a modular fashion. Each institution is represented by an EPN that can be executed independently or together with other institutions. The *individual behavior* for the robots is also represented by an EPN. While the institutions specify behaviors that have a *social nature*, i.e., they relate the robot to other robots in some way, the individual behavior specifies a set of basic behaviors that have exclusively an *individual nature*, i.e., they relate the robot with the surrounding environment. The composition of the individual behavior with a set of institutions will generate a robot controller.

We now present our formalized definition of institution:

Definition: An *Institution* I is a four-tuple $(Inst, initial_I, final_I, d_I)$ where:

- $Inst$ is an EPN;

- $initial_I$, $final_I \in Cdt$ are initial and final conditions for the execution of $Inst$;
- $d_I \in D$ is the associated deontic operator.

The EPN $Inst$ specifies the desired behavior that should be performed by the robot. This behavior is not always being executed, its start and finish are dictated by conditions $initial_I$ and $final_I$, which the robot verifies at each time step. Thus, we say that an institution I at each time step can be *active* or *idle*. Each institution also includes a deontic operator d_I which is used when combining it with the robot individual behavior and further institutions. Despite $Inst$ being designed by hand, institutions can be kept simple and further behavioral complexity can reached by composition, in a modular fashion.

The composition of the individual behavior with a set of institutions is non-trivial since concurrent execution of some of the institutions might be impossible or at least inadequate to the task the robot is carrying out. To guide this composition we introduce the following set of deontic operators.

Definition: The set D of deontic operators for IR institutions includes the following deontic operators: $\{AllowAll, StopInd, StopInst, StopAll\}$. Their corresponding definitions are as follows:

- $AllowAll$ implies that the associated institution can be executed concurrently with the individual behavior and all the other institutions;
- $StopInd$ implies that the associated institution cannot be executed concurrently with the individual behavior;
- $StopInst$ implies that the associated institution cannot be executed concurrently with other institutions;
- $StopAll$ implies that the associated institution cannot be executed concurrently with the individual behavior or other institutions.

Petri Nets (and thus EPN) can be represented in a hierarchical fashion, using two distinct layers. We consider that individual behavior and institutions are part of a lower layer and are represented by one macro place in the higher layer, as shown in Fig. 4.1. If a macro place is marked, the individual behavior or institution that it represents is active, otherwise it is idle. This allows us to compose our

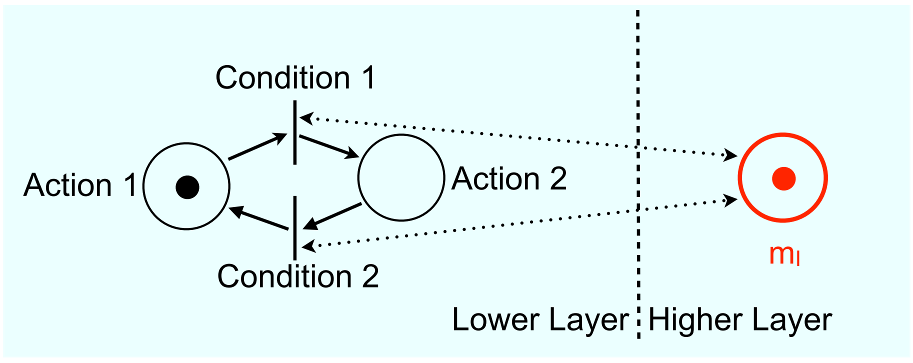


Figure 4.1: Hierarchical representation of an EPN in two layers. Dotted arcs represent two directional arcs, one from a transition to a place and one from a place to a transition. Left side: lower layer, EPN $Inst$ with conditions and actions associated to transitions and places. Right side: higher layer, macro place m_I in red.

institutions in the higher layer where relationships among the institutions and the individual behavior should be specified, while keeping relationships between actions and conditions separated in the lower layer. The composition procedure is explained in detail in (Pereira et al., 2011).

We can now define our Institutional Agent Controller that will guide the performance of our robots:

Definition: An *Institutional Agent Controller* (IAC) is an EPN resulting from the composition of an individual behavior Ind and a set of institutions $\{I_1, \ldots, I_n\}$ controlled by the deontic operators $d_{I_1}, \ldots, d_{I_n}$.

4.2.2 Modeling with Institutional Controllers

In our research we are interested in borrowing some key concepts from a multi-level probabilistic modeling methodology established for swarm robotic systems (Martinoli et al., 2004), that takes into account individual microscopic models based on the robots controllers and abstracts a macroscopic model of the dynamics of the whole team.

Following the intuition of using the robots controller as a starting point for our models, we will use Generalized Stochastic Petri Nets (GSPN) (Bause and Kritzinger, 2002) to construct our macroscopic model. In GSPN two types of transition are considered, immediate and timed. Timed transitions specify that a probabilistic time interval must pass before the transition can fire. If this time interval has a probabilistic exponential distribution with rate λ, we say that the transition is exponential with λ being called the transition rate. The marking process of a GSPN where all timed transitions are exponential is equivalent to a continuous time Markov chain (Bause and Kritzinger, 2002). This allows us to perform steady state analysis of the GSPN.

We consider a case study previously investigated in (Nembrini et al., 2002) and (Winfield et al., 2008), where a decentralized control algorithm is able to maintain a certain degree of spatial compactness of a robotic swarm (with N robots) in an unbounded arena using exclusively, as information at the robot level, the current number of wireless connections to the neighbors. The communication is local and its bounded range a parameter of the robotic system. Let X be the number of connections perceived by a robot. In the default state (defined as $forward$), the robot simply moves forward. If at any time the robot senses the loss of a connection and X falls below a threshold α (where $\alpha \in \{0, \ldots, N-1\}$), the robot assumes it is going in the wrong direction and switches to state $coherence$. In this state the robot performs a 180° turn in order to recover the lost connection. Upon recovering the lost connection, the robot performs a random turn and moves back to the default state. If the connection is not recovered, the robot simply moves to the default state. If an obstacle is detected the robot immediately switches to state $avoid$, where it performs obstacle avoidance for a given number of time steps, after which it returns to its previous state.

We designed an individual behavior Ind and institutions I_1 and I_2 in order for the robots to perform the task. These are displayed at the lower layer of Fig. 4.2. The composition of individual behavior and institutions is shown at the higher layer of Fig. 4.2. The final controller is the full EPN of Fig. 4.2 after merging the two layers.

We considered 40 robots in an unbounded arena performing the task over 10 000 seconds. We fix the connection threshold to one single value, $\alpha = 15$, and set the communication radius of the robots to 0.7 m. We performed 100 runs of a realistic simulation using *Webots*.

When applying the proposed modeling methodology to the institutional robotics approach we are mainly interested in studying the relationships between different behaviors specified in the higher layer of the IAC. However, in some cases,

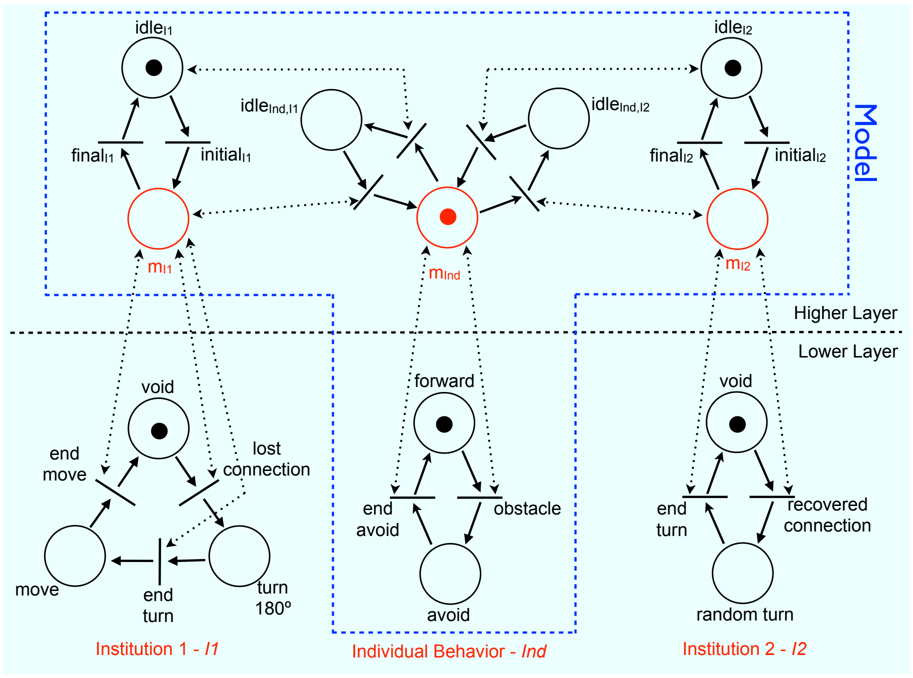


Figure 4.2: IAC for wireless connected swarm. Lower layer: EPNs for individual behavior Ind and institutions I_1 and I_2. Higher layer: composition of individual behavior and institutions. PN structure for GSPN model encapsulated in blue box.

details about implementation of behaviors might also be of interest. By using the two layers of the IAC we can select sections of the EPN that are of interest for a given model. For our proposed model, the higher layer would suffice if only states $forward$ and $coherence$ were considered (with $coherence$ corresponding to markings where the macro place for institution I_1 is marked). However, to also consider state $avoid$ we need the lower layer implementation of the individual behavior, in order to make the distinction between $avoid$ and $forward$. The GSPN structure for our model is presented in Fig. 4.2 as the section of the IAC encapsulated by the blue box.

The only immediate transitions in the model are those that are not associated with any condition. These correspond to the control transitions added during composition of behaviors, and are the transitions linking the macro place of the individual behavior m_{ind} with idle places $idle_{ind,I1}$ and $idle_{ind,I2}$. The remaining transitions in the model are timed and their transition rates need to be estimated. These correspond to conditions $obstacle$, *end avoid*, and the initial and final conditions for both institutions. Given that our goal is only to establish that the IAC structure can be used as a GSPN model, we choose to estimate the transition rates directly from data gathered during realistic simulations. The transition rates are calculated separately for each number of connections ($k = 0, \dots, 40$). We do this by counting the number of time steps the input places of transition i are marked ($t_{i,k}$) and the number of times transition i fires ($f_{i,k}$), for all robots in all runs, while the number of connections of the robot is k. These are then averaged by the total number of time steps, the number of robots and of runs (averages represented by $\overline{t_{i,k}}$ and $\overline{f_{i,k}}$). The rate of transition i with k connections is calculated as $\lambda_{i,k} = \overline{f_{i,k}}/\overline{t_{i,k}}$.

For each number of connections, we perform steady state analysis on our GSPN model with rates $\lambda_{i,k}$ ($i = 1, \dots, 6$). This analysis gives us the steady state probability for each tangible marking in our GSPN model. As before, we consider that state $avoid$ corresponds to markings where the individual behavior is active and the $avoid$ place is marked, state $coherence$ corresponds to markings where institution I_1 is active, and all other markings correspond to state $forward$. By summing the probabilities for all markings corresponding to each state we obtain the desired state distribution model.

The model results are displayed in Fig. 4.3-(a). They show how the probability of being in each state varies with the number of connections. In Fig. 4.3-(b), we display the state distribution from our simulations results. The absolute error between model and simulation results is display in Fig. 4.4. We can observe an almost perfect matching between the two results. This is to be expected since our estimation of transition rates comes directly from the data gathered during the simulations. The larger error in Fig. 4.4 comes from the extremely low number of time steps and transition fires for robots with 38 connections. This affects the calculation of the correct rate and thus generates a bigger error in the model. We conclude that with a correct estimation of transition rates, our IAC provides a good structure for the generation of macroscopic models.

4.3 Summary and Future Work

Using the EPN structure of the IAC designed for the case study, we were able to construct a GSPN model for overall state distribution of the system. By using data gathered from realistic simulations in order to estimate the transition rates

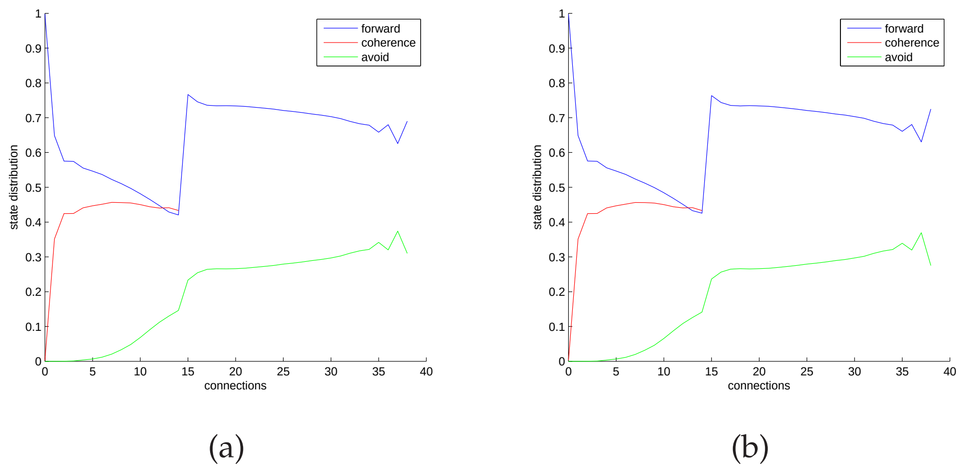


Figure 4.3: (a) State distribution predicted from GSPN model; (b) State distribution from simulation results.

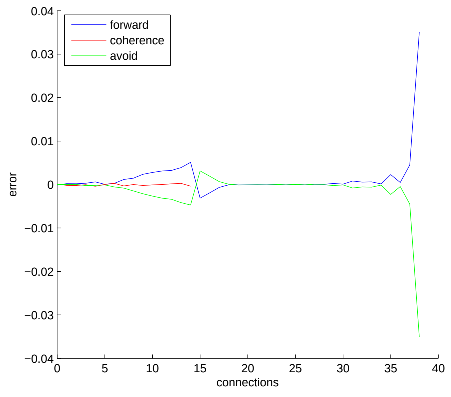


Figure 4.4: Absolute error between model and simulation results.

necessary for our GSPN model, we were able to observe a very good agreement between model predictions and simulation results. We can conclude that with a correct estimation of transition rates, our IAC provides a good structure for the generation of macroscopic models.

In the future we intend to further improve our GSPN models, for instance, with estimation of transition rates directly computed using geometrical considerations of the scenario rather than on data gathered in simulation. An alternative algorithm for this case study, considering the sharing of neighborhood information among robots, is presented in (Nembrini et al., 2002). We intend to design an

IAC for this algorithm and apply our modeling methodology, in order to investigate situations where the microscopic-to-macroscopic (or individual-to-swarm) mapping might be less straightforward to capture accurately because of the additional complexity of the coordination algorithm.

Appendix A

Binomial heap implementation for fast multi-cellular stochastic simulation

A.1 Stochastic simulation algorithm

Simulating the stochastic dynamics of the molecular reaction events within the cell, and the events leading to changes in cell population size and composition is a computational challenge. In this study, we propose a novel implementation of the Gillespie stochastic simulation algorithm (Gillespie, 1976) that keeps track of the multilevel granularity of the system and is computationally efficient. The core of the algorithm is the storage of events in a binomial heap (Chapter 19 in (Cormen et al., 2001)), a data structure which was developed by Jean Vuillemin in the mid 1970s (Vuillemin, 1978). Binomial heaps have been characterised as a practical and nearly optimal priority queue implementation (Brown, 1978), allowing the critical operation of finding the next event to be fired from a dynamically changing set of events in time proportion to the *logarithm* of the number of events, not to the number of events itself. In addition, the computational time taken for operations on binomial heaps are not very sensitive to the priority distribution of events, as demonstrated with a discrete event simulation model (Jones, 1986). In terms of computational memory usage, a binomial heap data structure allows for an efficient and flexible allocation of memory. This is in contrast to the binary heap data structure (Chapter 6 in (Cormen et al., 2001)), used previously to implement Gillespie like algorithms (Gibson and Bruck, 2000; Stamatakis and Zygourakis, 2010), and requiring a contiguous allocation of memory blocks to operate. In summary, the binomial heap serves as an ideal data structure to implement the priority queue of a dynamically varying set of reactions of our stochastic simulation

A.2 Binomial tree

A binomial heap consists of a collection of binomial trees. In this section we define a binomial tree and prove some of its important properties. Consequently, we define a binomial heap and illustrate its representation in our stochastic simulation.
A binomial tree B_k is an ordered tree (Appendix B in (Cormen et al., 2001)) defined recursively. The tree B_0 consists of a single node, and B_k consists of two binomial trees B_{k-1} that are linked together i.e., the root of one is the leftmost child of the root of the other (Figure A.1a). The binomial trees B_0 through B_3 are indicated in Figure A.1b.

Lemma A.2.1 *The properties of of the binomial tree B_k.*

1. *The tree has 2^k nodes.*
2. *The height of the tree is k.*
3. *There are exactly $\binom{n}{k}$ nodes at depth i for $i = 0 \ldots k$.*
4. *The root node has degree k, which is greater than that of any other node in the tree.*

Proof The properties outlined above can be proved by induction on k. For each property (Lemma A.2.1), the basis is the binomial tree B_0. Each of the properties can easily be verified for B_0. The inductive step assumes that Lemma A.2.1 holds for B_{k-1}

1. The binomial tree B_k consists of two copies of B_{k-1} linked together. Consequently, B_k has $2^{k-1} + 2^{k-1} = 2^k$ nodes.
2. Consequent to the manner in which the two B_{k-1} trees are linked together to form B_k, the maximum depth of a node in B_k is one greater than the maximum depth of a node in B_{k-1}. Therefore by the inductive hypothesis, the maximum depth of B_k is $(k-1)+1 = k$.

3. Let us denote $D(k,i)$ to be be the number of nodes at depth i of binomial tree B_k. Consequent to the manner in which which the two copies of B_{k-1} are linked together to form B_k, a node at depth i in B_{k-1} appears once at depth i in B_k and once at depth $i+1$. So the number of nodes at depth i in B_k is the sum of the number of nodes at depth $i-1$ in B_{k-1} and the number of nodes at depth $i-2$ in B_{k-1}.

   Following this, the number of node at depth i of binomial tree B_k can be expressed as,
   $D(k,i) = D(k-1,i) + D(k-1,i-1)$
   $D(k,i) = \binom{k-1}{i} + \binom{k-1}{i-1}$ (by the inductive hypothesis)
   When solved, the sum of the two binomial coefficients reduces to,
   $D(k,i) = \binom{k}{i}$

4. The only node with greater degree in B_k than in B_{k-1} is the root node of B_k which has one more child than in B_{k-1}. Since the root of B_{k-1} has degree $k-1$, the root of B_k has degree k. ∎

A.3 Binomial heap

A binomial heap H is defined as a collections of binomial trees that satisfies the following properties.

1. Each binomial tree H obeys the ***min-heap property***. The key (e.g. reaction waiting time) of a node is greater than or equal that of its parent. Consequently, the root of each tree contains the lowest reaction waiting time in that tree. Such a tree is referred to as ***min-heap-ordered***.

2. For any non-negative integer k, there is at most one binomial tree B_k in H.

The second property implies that a binomial heap H containing the waiting time of n reactions will consist of at most $\lfloor \lg n \rfloor + 1$ binomial trees. This can be understood by observing that the binary representation of n has $\lfloor \lg n \rfloor + 1$ bits. say $\langle b_{\lfloor \lg n \rfloor}, b_{\lfloor \lg n \rfloor - 1}, \ldots b_0 \rangle$, so that $n = \sum_{i=0}^{\lfloor \lg n \rfloor} b_i 2^i$. Consequent to property 1 of Lemma A.2.1, the binomial tree B_i appears in H if and only if bit $b_i = 1$. Therefore, the binomial heap H can contain at most $\lfloor \lg n \rfloor + 1$ binomial trees. Figure A.2a, shows a binomial heap H with 11 nodes. The binary representation of 11 is 1011

, and H consists of min-heap-ordered binomial trees B_3, B_1, and $B0$, having 8, 2, and 1 nodes respectively, for a total of 11 nodes.

A.3.1 Binomial heap representation

In our stochastic simulation, each binomial tree within the binomial heap is stored utilising the left-child, right-sibling representation (Chapter 10 in (Cormen et al., 2001)). Each node of the tree has a key (the waiting time for a reaction) and a pointer to information about the reaction. In addition, each node t contains pointers $p\,[t]$ to its parent, $child\,[t]$ to its leftmost child, and $sibling\,[t]$ to the sibling of t at its immediate right (Figure A.2b). Additionally, each node t also contains $degree\,[t]$ i.e., the number of its children. As indicated in Figure A.2b, the roots of all the binomial trees in the heap are organised in a linked list, which is further referred to as the ***root list***. The degrees of the roots strictly increase as we traverse the root list. Consequent to the second binomial heap property, the degrees of the roots of a n-node binomial heap are a subset of $\{0, 1, \ldots \lfloor \lg n \rfloor\}$. The $sibling$ field has a different meaning for roots than for non-roots. If t is in the root list, $sibling\,[t]$ points to the next root in the list.

A.3.2 Operations performed on binomial heaps

During our stochastic simulation, events pertaining to molecular reactions, and changes in Th cell population and composition are added, deleted, or updated (regeneration of reaction waiting time) in the heap. We now outline the corresponding operations performed on the heap, and illustrate their working with an example binomial heap.

MAKE-BINOMIAL-HEAP to create a new binomial heap: A given binomial heap H is accessed with $head\,[H]$, a pointer to the ïňĄrst root in the root list of H. In order to make an empty binomial heap, the MAKE-BINOMIAL-HEAP procedure allocates and returns returns an object H, with $head\,[H] = NIL$, in a constant amount of time.

BINOMIAL-HEAP-MINIMUM to find the next reaction to occur: The procedure BINOMIAL-HEAP-MINIMUM returns the pointer to the node with the lowest reaction waiting time in an n-node binomial heap H. Since the binomial heap consists of a collection of min-heap-ordered binomial trees, the reaction with the lowest waiting time must reside in one of the root nodes. This procedure checks all the root nodes of the binomial heap for the reaction with the lowest waiting time. Because the root nodes of the binomial heap number at most $\lfloor \lg n \rfloor + 1$, the next reaction to occur is found in time $O(\lg n)$.

BINOMIAL-HEAP-UNION to unite two binomial heaps: The procedure to unite two binomial heaps, say $H1$ and $H2$ is used by operations to insert, delete and update reactions to the heap. It involves two phases. The first phase merges the root lists of binomial heaps $H1$ and $H2$ into a single linked list H that is sorted by degree in a monotonically increasing order (Figure A.3a and b). In the resulting heap H, there may be as many as two roots (but no more) of each degree. Consequently, the second phase of the union operation links roots of equal degree until at most one root remains of each degree. In Figure A.3b and c, we demonstrate the linking of the two binomial trees B_1 in H, resulting in B_2. Subsequently, two binomial trees B_2 are now present in the heap, which are linked in the next step to form B_3 (Figure A.3c and d), and the operation can now be terminated. The procedure has been shown to have a running time of $O(\lg n)$ (Chapter 19 in (Cormen et al., 2001)).

BINOMIAL-HEAP-INSERT to insert a new reaction into the heap: During the course of our stochastic simulation, new reactions are added to the priority queue depending on the nature of the reaction that has just occurred (see dependency table). The insert procedure simply makes a one-node binomial heap $H^{'}$ in constant time and unites it with the n-node binomial heap H by a call to BINOMIAL-HEAP-UNION, in $O(\lg n)$ time. To demonstrate this procedure, consider a 9-node binomial heap H (Figure A.4a). The node to be inserted is in 1-node heap $H^{'}$ which is merged with H (Figure A.4b). The operation of uniting two binomial heaps in now performed on H until at most one root remains of each degree (Figure A.4c and d).

BINOMIAL-HEAP-EXTRACT to extract the next reaction to occur from the heap: Upon the occurrence of a reaction, we remove the node representing it from the heap. The extraction procedure as demonstrated on an input binomial heap H (Figure A.5a), starts by removing the root t with the minimum key from the root list of H (Figure A.5b). The Figure A.5c, shows that by reversing the list of t's children, we have a binomial heap $H^{'}$ that contains every node in t's tree except for t itself. Since t's tree was removed from H, the binomial heap (Figure A.5d), that results from uniting H and $H^{'}$ contains every node that was present in H except for t.

BINOMIAL-HEAP-UPDATE to update the waiting time of a reaction in the heap: Consequent to the occurrence of certain events (listed in dependency table), the waiting time of an event in the binomial heap may have to be regenerated, and its position in the heap updated. When the newly generated waiting time is no greater than the current value, the following procedure is employed to reposition the node. As shown in Figure A.6, the procedure repositions the associated node by "bubbling it up" the heap. As an example consider a binomial heap H with a node t whose waiting time has to be updated (Figure A.6a). After ensuring that the new waiting time is in fact no greater than the current value and then assigning the new value to t, the procedure goes up the tree, with c initially pointing to node t. In each iteration, $key\,[c]$ is compared to the key of c's parent p. If c is the root or $key\,[c] \geq key\,[p]$, the binomial tree is now min-heap-ordered. Otherwise, node c violates min-heap ordering, and therefore its key is exchanged with the key of its parent p, along with information about the reaction. The procedure then updates c to p (Figure A.6b and c), going up one level in the tree, and proceeds with the next iteration.

The above procedure operates on the condition that the updated key is no greater than the current key. In cases when this condition is not satisfied, the corresponding node t is deleted from the heap (see procedure BINOMIAL-HEAP-DELETE), and a new node with the regenerated waiting time is inserted into heap H.

BINOMIAL-HEAP-DELETE to delete a reaction from the heap. The procedure to delete a node x from the heap H is performed by first assigning $key\,[x]$ to $-\infty$. Following the update to $key\,[x]$, the node x is bubbled up to the root of the tree by a call to BINOMIAL-HEAP-UPDATE. This root is then removed from H by a call to the BINOMIAL-HEAP-EXTRACT procedure.

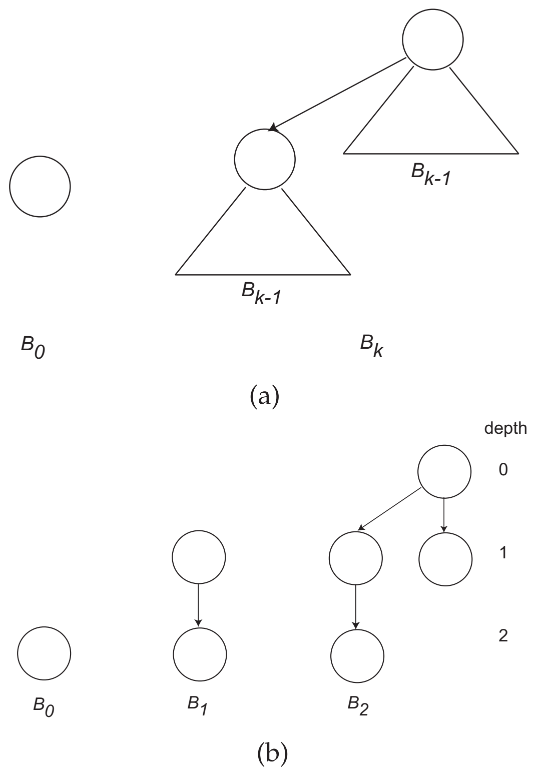
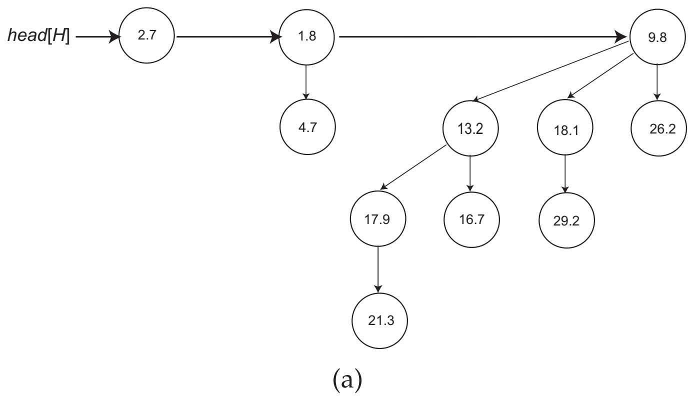


Figure A.1: **Collection of binomial trees.** (a) The recursive deïňĄnition of the binomial tree B_k . The triangles adjoining the nodes represent rooted sub-trees. (b) The binomial trees B_0 through B_3. Depth of nodes of B_3 are shown.



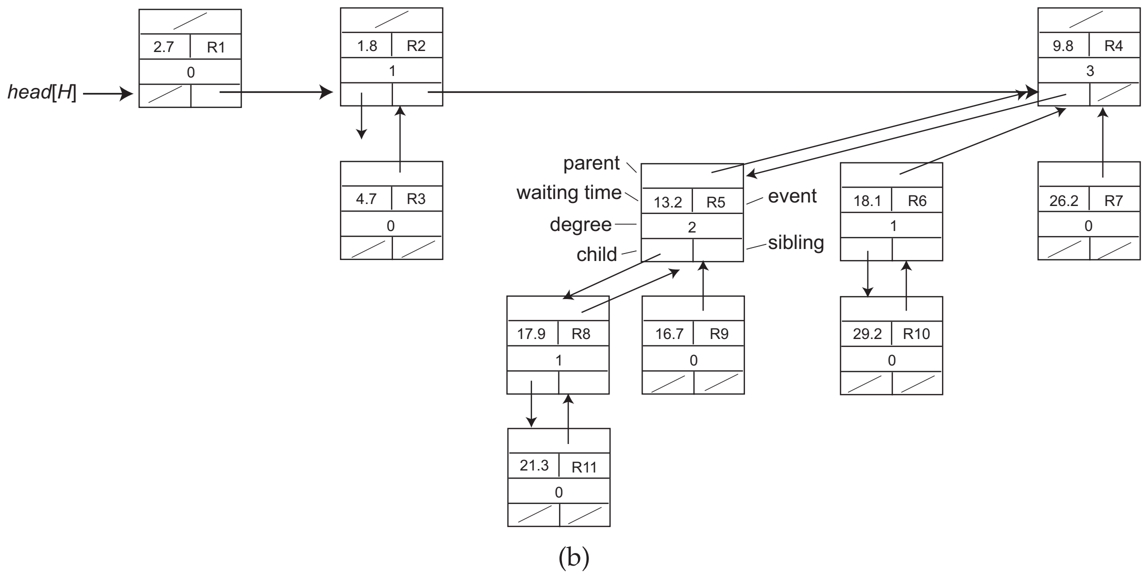


Figure A.2: **Representation of a binomial heap.** A binomial heap H with 11 nodes. (a) The heap H consists of binomial trees B_0, B_1 and B_3, which have 1, 2, and 8 nodes respectively, totalling 11 nodes. Since each binomial tree is min-heap-ordered, the waiting time of any node is no less than that of its parent. Also indicated is the root list, a linked list of roots in order of increasing degree. (b) The representation of the binomial heap H, as implemented in our stochastic simulation.

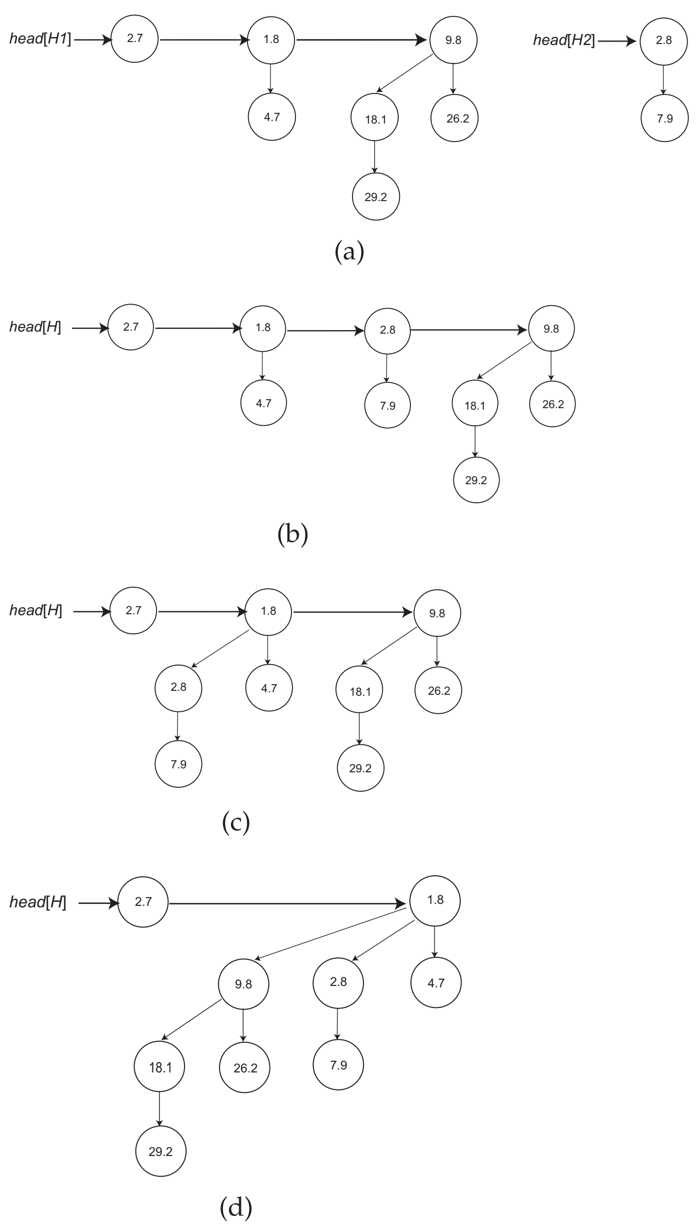


Figure A.3: **Uniting two binomial heaps.** (a) Binomial heaps $H1$ and $H2$ that are to be united. The heap $H1$ consists of binomial trees B_0, B_1 and B_2, while $H2$ consists of B_2. (b) A heap H from merging of $H1$ and $H2$ has two binomial trees B_1. (c) The union of the two trees B_1 in H, into a binomial tree B_2. (d) Proceeding along the root list, the union of two binomial trees B_2 results in B_3. As shown, the heap H now has at most one root of each degree.

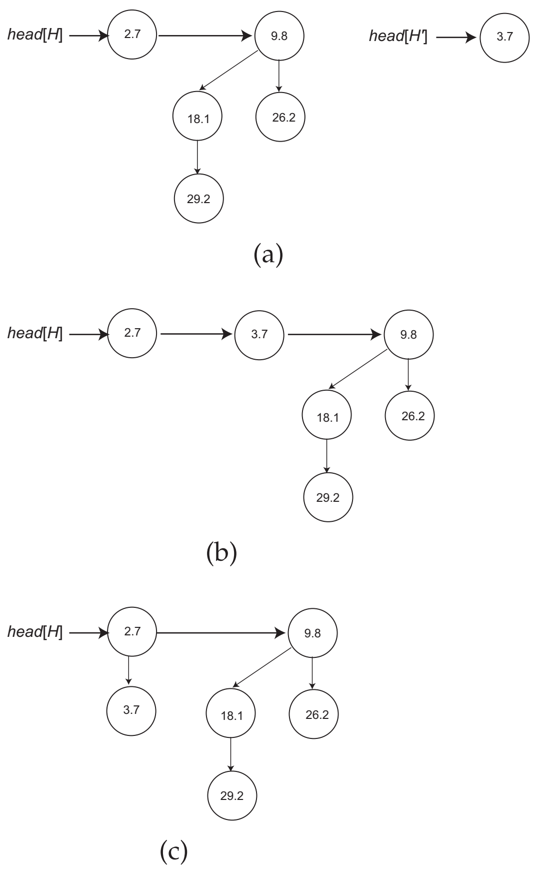


Figure A.4: **Insertion of new event into binomial heap.** (a) A binomial heap H with 5 nodes and the heap $H^{'}$ consisting of the new node to be inserted. (b) Merging of the two heaps results in H having two binomial trees B_0. (c) The two trees B_0 are united as shown, resulting in binomial tree B_1. The heap H now has at most one root of each degree.

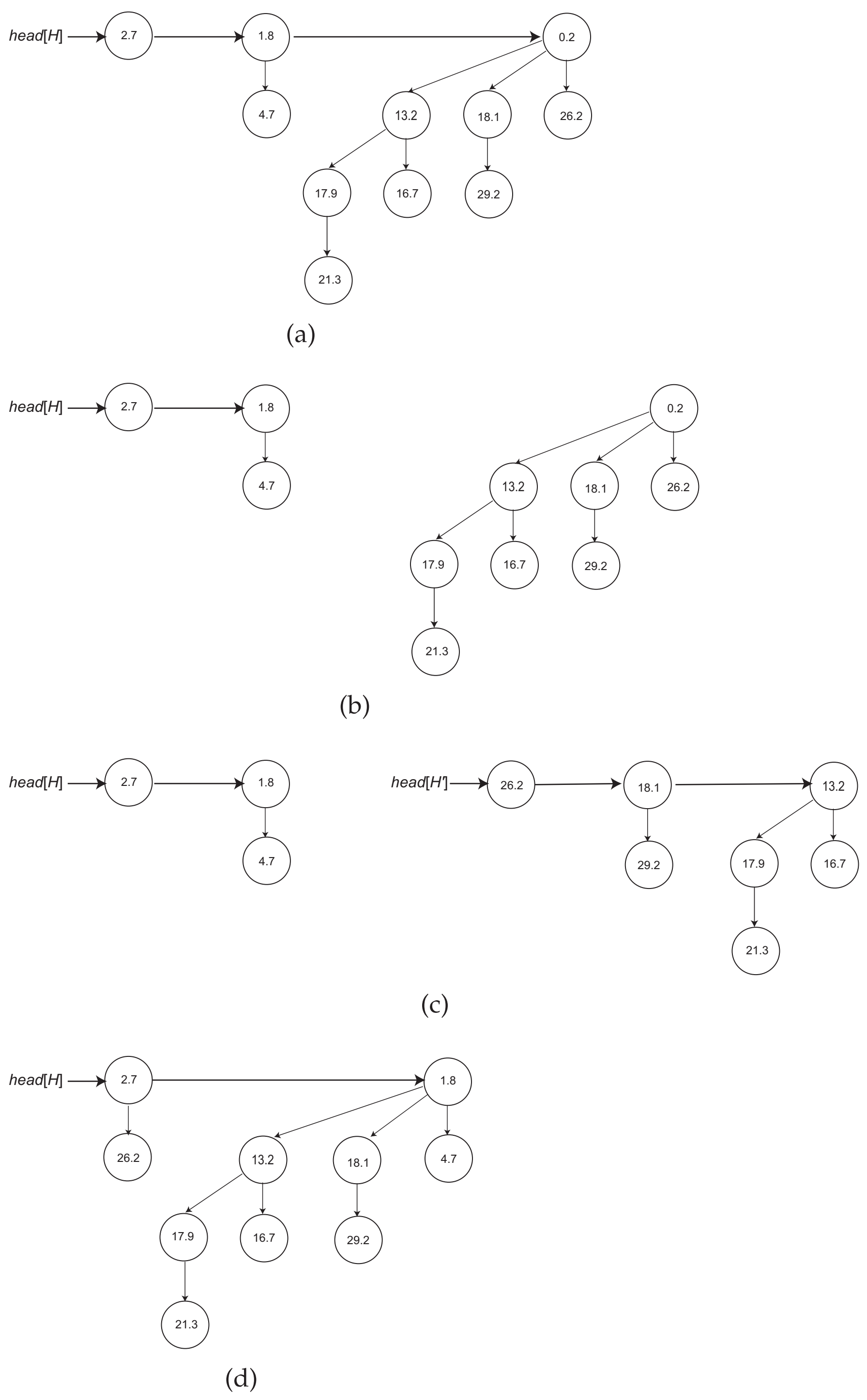
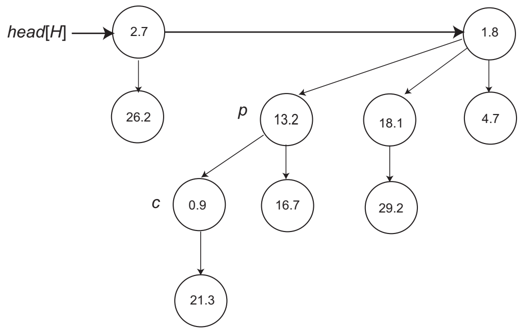
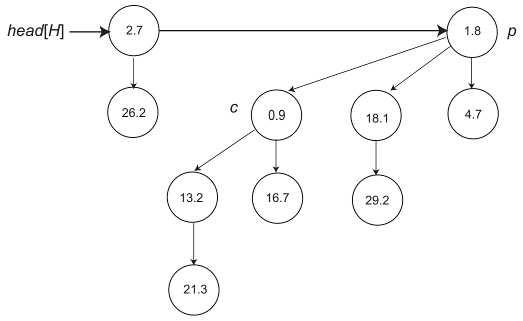


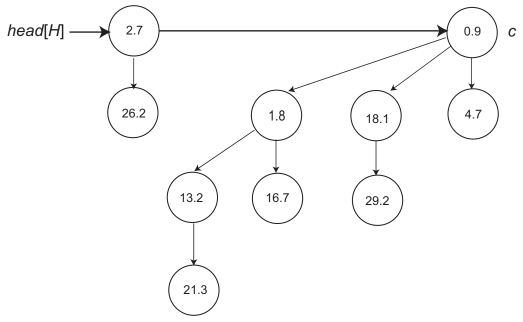
Figure A.5: **Extraction of next event from binomial heap.** (a) A binomial heap H with 11 nodes. The next reaction to be executed is associated with the root node t of B_3. (b) The root t associated with the next reaction to occur, is removed from the heap H. (c). The list of t's children is reversed, to form another binomial heap $H^{'}$.(d) The result of uniting the two binomial heaps, H and $H^{'}$ are united.



(a)



(b)



(c)

Figure A.6: **Repositioning of event in binomial heap after update in waiting time.** (a) A binomial heap H, with a node c with an updated waiting time. Consequently, the heap H is not min-heap-ordered. (b) The result of exchanging nodes c and p in H, but the min-heap-order of H continues to be violated.(c) The result of another exchange of c and p The min-heap-order of H is now satisfied.

Appendix B

Parameters for multi-cellular stochastic simulation

Table B.1: **Event parameters of Th cell gene regulatory network.**

Component	Average waiting time
CD28, IFNBR, IFNGR, IL2R, IL4R, IL6R, IL10R, IL12R, IL15R, IL21R, IL23R, IL27R, TCR and TGFBR	$0.05\ h$
NFAT, STAT1, STAT3, STAT4, STAT5, STAT6, SMAD3, RUNX3, NFKB and IKB	$1\ h$
IL12RB1, IL12RB2, IL2RA, TBET, GATA3, FOXP3, IRF1, RORGT, IL2*, IFNG**, IL17**, IL4**, IL10**, IL21**, IL23** and TGFB**	$20\ h$

* = cytokine component is not activated in G0

** = cytokine component is not activated in G0 and G1

Average waiting times for three different classes of components of the cell logical network. The waiting times apply to both unitary increments and decrements of the component level.

Table B.2: **Event parameters of Th cell cycle.**

Parameters	Description	Value
σ_{G0}	Residual time in quiescent phase	$46.5\ h$
σ_{G1}	Residual time in first growth phase	$0.7\ h$
σ_{G1}^{R}	Residual time to revert to quiescent phase	$72\ h$
τ_S	Average time to synthesise DNA	$6\ h$
τ_{G2M}	Average time in second growth stage and mitosis	$2\ h$
τ_D	Average lifespan of a Th cell	$67\ h$

Mean waiting times for Th cell cycle transitions, and cell apoptosis.

Table B.3: **Event parameters of Th cell specific cytokine environment**

Parameters	Description	Value
k_p	Cytokine production rate	$10^{-1}\ nU.h^{-1}.cell^{-1}$
k_{in}	Internalisation rate, when receptor component at level 1	$10^{-5}\ h.cell^{-1}$
k_{in2}	Internalisation rate, when receptor component at level 2 (IL-2)	$10^{-3}\ h.cell^{-1}$
k_d	Degradation rate	$0.138 \times 10^{-2}\ h^{-1}$
k_{Th}	Additional cytokine at local pool	$200\ nU.cell$
θ	Cytokine receptor threshold	$500\ nU$

Parameters of cytokines produced by Antigen presenting cells, and consumed by Th cells. The cytokines simulated are IFNγ, IL-2, IL-4, IL-10, IL-21, IL-23 and TGFβ.

Table B.4: **Event parameters of Antigen presenting cell specific cytokine environment**

Parameters	Description	Value
k_p	Cytokine production rate	10^{-2} $nU.h^{-1}.cell^{-1}$
k_{in}	Internalisation rate, when receptor component at level 1	10^{-5} $h.cell^{-1}$
k_d	Degradation rate	0.138×10^{-2} h^{-1}
k_A	Additional cytokine at local pool	200 nU
θ	Cytokine receptor threshold	500 nU

Parameters of cytokines produced by Antigen presenting cells, and consumed by Th cells. The cytokines simulated are IFNβ, IL-6, IL-12, IL-15 and IL-27.

Table B.5: Description of model parameters of Th cells and conjugations with Antigen Presenting Cells (APC).

Parameters	Description	Value
V	Volume of medium	10^{-9} $Liters$
k_{on}	Rate of contact between Th cells and APCs	3.6×10^{-10} $Liters.h^{-1}.cell^{-1}$
k_{off}	Average time to unproductive dissociation	5 h
k_{sc}	Average time cell stays strongly conjugated	2 h

Appendix C

Pseudo-code of decentralized and institution implementation

Algorithm C.1 Simulation of decentralised decision making

Parameters:

m Number of slots at assembly site

n Number of different types of components at the building site

T Number of agents

SSA Proportion of short sighted agents in the population

γ_{SSA} Discount factor of short sighted agents

γ_{FSA} Discount factor of far sighted agents

RC The collective reward for piece completion

E Number of time-steps to evaluate population

```
 1: Initialise the agents with the given discount factors
 2: Initialise the assembly site, $\forall i : C_i = \phi$
    {Simulation steps}
 3: for $i = 1 \to E$ do
 4:    {Randomly select one of the $T$ agents}
 5:    $a \leftarrow RAND(T)$
       {Randomly select one of the $n$ component types}
 6:    $F_a \leftarrow RAND(n)$
       {The subroutine DECENTZ-POLICY returns where the component should
       be placed}
 7:    $j \leftarrow DECENTZ - POLICY(C_m, C_{m-1} \dots C_1, \gamma_a, RC, F_a, n)$
 8:    Place component in slot $j$ and accumulate immediate reward
 9:    if Assembly site has a completed piece then
10:       Remove completed piece from assembly site
11:       Update the total number of completed pieces
12:       Distribute collective reward $RC$ amongst contributing agents
13:    end if
14:    Check for availability of at least one free slot. If none available, empty all
       the slots at the assembly site.
15: end for
16: return  Number of pieces completed
```

Algorithm C.2 DECENTZ-POLICY: Function to determine the slot where component is to be placed

Parameters:

C_m Component at slot m

C_{m-1} Component at slot $m-1$

$\vdots$

C_1 Component at slot 1

γ Discount factor of focal agent

RC Collective reward F Component brought by agent to assembly site

n Number of different types of components at the building site

```
1: {Call subroutine to compute the prospect vector ($P$) for the $m$ building slots
   at the assembly site, and component type $F$}
2: $P \leftarrow PROSPECT(C_m, C_{m-1} \ldots C_1, F, n)$
3: **for** $j = 1 \rightarrow m$ **do**
4:     $u_j \leftarrow j + P_j(RC \times \gamma)$
5: **end for**
   {Return slot with maximum utility that is free}
6: **return** $(\arg\max_{x \in \{1 \ldots m\}}(u_x \wedge C_x = \phi))$
```

Algorithm C.3 PROSPECT: Function to compute the prospect vector (P) for the assembly site. For each slot j, P_j indicates if placement of component F in slot j would allow piece completion in the future. $P_j = 1$ if piece completion possible, else $P_j = 0$

Parameters:

C_m Component at slot m

C_{m-1} Component at slot $m-1$

$\vdots$

C_1 Component at slot 1

F Component brought by agent to assembly site

n Number of different types of components at the building site

```
1:  {Evaluation of prospect at each of the slots}
2:  for $slot = m \rightarrow 1$ do
3:      $P_{slot} \leftarrow 0$
4:      if $C_{slot} \neq \phi$ then
5:          {Current slot is not available. We set its prospect to 0 and move onto next
            slot}
6:          Continue to next $slot$
7:      end if
        {We now check if succeeding components can be placed to complete the
        piece}
8:      if Components $F+1 \rightarrow n$ can not be placed after the component $F$ then
9:          {Incorrect components are present. The piece cannot be completed.}
10:         Continue to next $slot$
11:     end if
        {We now check if preceding components can be placed to complete the
        piece}
12:     if Components $1 \rightarrow F-1$ can not be placed before the component $F$ then
13:         {Incorrect components are present. The piece cannot be completed.}
14:         Continue to next $slot$
15:     end if
16:     $P_{slot} \leftarrow 1$
17: end for
18: return $P$
```

Algorithm C.4 Simulation of Institution decision making

Parameters:

m Number of slots at assembly site

n Number of different types of components at the building site

T Number of agents

AF Assembler fee

RC The collective reward for piece completion

E Number of time-steps to evaluate population

```
 1: Initialise the set of waiting agents $W = \phi$
 2: Initialise the assembly site, $\forall i : C_i = \phi$
    {Initialisation of ordered list $Q$ used by assembler to store components needed
    for piece completion}
 3: $Q \leftarrow \phi$
    {Simulation steps}
 4: **for** $i = 1 \rightarrow E$ **do**
 5:    {Search the set of waiting agents $W$ for a required piece, i.e. one that is not
       already in the list $Q$}
 6:    **if** $\exists i \in W : F_i \notin Q$ **then**
 7:       $W \leftarrow W - i$
 8:       $Q \leftarrow Q + F_i$
 9:       $F_i \leftarrow \phi$
10:       Continue to next stimulation step
11:    **end if**
12:    {Randomly select one of the available agents}
13:    $a \leftarrow RAND(T - length(W))$
       {Randomly select one of the $n$ component types}
14:    $F_a \leftarrow RAND(n)$
       {Check if component $F_a$ is already in set $Q$, i.e. is it needed for piece com-
       pletion}
15:    **if** $F_a \notin Q$ **then**
16:       $Q \leftarrow Q + F_a$
17:       $F_a \leftarrow \phi$
18:    **else**
19:       {Add the agent to the set of agents waiting at the assembler}
20:       $W \leftarrow W + a$
21:       Continue to next stimulation step
22:    **end if**
       {The subroutine INST-POLICY determines the action of the assembler}
23:    Call subroutine $INST - POLICY(Q, m, n)$
24: **end for**
25: **return** Number of pieces completed
```

Algorithm C.5 INST-POLICY: Function to determine the action of the assembler
Parameters:
Q Ordered list components
m Number of slots at assembly site
n Number of different types of components at the building site

```
1: if $Q = \{1, 2 \ldots n\}$ then
2:     {Empty $Q$ into the assembly site in slots $\{C_m, C_{m-1} \ldots C_{m-n+1}\}$}
3:     $\{C_m, C_{m-1} \ldots C_{m-n+1}\} \leftarrow Q$
4:     $Q \leftarrow \phi$
5:     Simultaneously award contributing agents their individual 'immediate' re-
       ward ($RI$ =slot index) and the common collective reward ($RC - AF$)
6:     Increment number of pieces completed
7:     Empty the assembler i.e., remove the completed piece
8: end if
```

Bibliography

Bause, F. and Kritzinger, P. S. (2002). *Stochastic Petri Nets - An Introduction to the Theory*. Friedr. Vieweg &Sohn Verlag, Braunschweig/Wiesbaden (Germany), second edition.

Brown, M. (1978). Implementation and analysis of binomial queue algorithms. *SIAM Journal on Computing*, 7:298–319.

Butcher, J. (2003). *Numerical methods for ordinary differential equations*, chapter 23. John Wiley & Sons, second edition.

Cassandras, C. G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer, second edition.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, second edition.

Correll, N. and Martinoli, A. (2011). Modeling and designing self-organized aggregation in a swarm of miniature robots. *International Journal of Robotics Research*, 30(5):615–626.

Evans, M., Hastings, N., and Peacock, B. (2000). *Statistical Distributions*, chapter 27, pages 134–136. John Wiley & Sons, third edition.

Garg, A., Mohanram, K., Cara, A., DeMicheli, D., and Xenarios, I. (2009). A network model for the control of the differentiation process in Th cells. *Bioinformatics*, 25:101–109.

Garg, A., Xenarios, I., Mendoza, L., and DeMicheli, D. (2007). An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments. In *Proceedings of the 11th annual international conference on Research in computational molecular biology*, pages 62–76. Springer-Verlag.

Gibson, M. and Bruck, J. (2000). Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, 104:1876–1889.

Gillespie, D. (1976). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434.

Harrington, L., Hatton, R., Mangan, P., Turner, H., Murphy, T., Murphy, K., and Weaver, C. (2005). Interleukin 17-producing CD4+ effector T cells develop via a lineage distinct from the T helper type 1 and 2 lineages. *Nat Immunol*, 6:1123–1132.

Hauert, S., Zufferey, J., and Floreano, D. (2009). Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32.

Hori, S., Nomura, T., and Sakaguchi, S. (2003). Control of regulatory T cell development by the transcription factor Foxp3. *Science*, 299:1057–1061.

Hosken, N., Shibuya, K., Heath, A., Murphy, K., and O'Garra, A. (1995). The effect of antigen dose on CD4$^+$ t helper cell phenotype development in a t cell receptor-$\alpha\beta$-transgenic model. *The Journal of experimental medicine*, 182:1579–84.

Jones, D. (1986). An empirical comparison of priority-queue and event-set implementations. *Communications of the ACM*, 29:300–311.

Leon, K., Lage, A., and Carneiro, J. (2003). Tolerance and immunity in a mathematical model of t-cell mediated suppression. *J. Theor. Biol.*, 225:107–126.

Martinoli, A., Easton, K., and Agassounon, W. (2004). Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *Int. Journal of Robotics Research*, 23(4):415–436.

Mendoza, L. (2006). A network model for the control of the differentiation process in Th cells. *Biosystems*, 84:101–114.

Mermoud, G., Upadhyay, U., Evans, W. C., and Martinoli, A. (2010). Top-Down vs Bottom-Up Model-Based Methodologies for Distributed Control : A Comparative Experimental Study. In *12th International Symposium on Experimental Robotics*, New Delhi, India. Springer Tracts in Advanced Robotics.

Mosmann, T. and Coffman, R. (1989). Th1 and Th2 cells: different patterns of lymphokine secretion lead to different functional properties. *Annu Rev Immunol*, 7:145–173.

Murphy, E., Shibuya, K., Hosken, N., Openshaw, P., Maino, V., Davis, K., Murphy, K., and O'Garra, A. (1996). Reversibility of t helper 1 and 2 populations is lost after long-term stimulation. *The Journal of experimental medicine*, 183:901–13.

Naldi, A., Carneiro, J., Chaouiya, C., and Thieffry, D. (2010). Diversity and plasticity of Th cell types predicted from regulatory network modelling. *PLoS Comput Biol*.

Nembrini, J., Winfield, A. F. T., and Melhuish, C. (2002). Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots. *Proceedings of the seventh international conference on simulation of adaptive behavior on From animals to animats*, pages 273–282.

O'Grady, R., Groß, R., Christensen, A., and Dorigo, M. (2010). Self-assembly strategies in a group of autonomous mobile robots. *Autonomous Robots*, 28:439–455. 10.1007/s10514-010-9177-0.

Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C., and Tummolini, L. (2004). Coordination artifacts: Environment-based coordination for intelligent agents. In *3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 286–293, New York, NY, USA.

Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434.

Park, H., Li, Z., Yang, X., Chang, S., Nurieva, R., Wang, Y., Wang, Y., Hood, L., Zhu, Y., Tian, Q., and Dong, C. (2005). A distinct lineage of CD4 T cells regulates tissue inflammation by producing interleukin 17. *Nat Immunol*, 6:1133–1141.

Parker, C., Zhang, H., and Kube, C. (2003). Blind bulldozing: multiple robot nest construction. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 2010–2015.

Pereira, J. N., Silva, P., Lima, P. U., and Martinoli, A. (2011). Formalizing Institutions as Executable Petri Nets for Distributed Robotic Systems. In Lenaerts, T., Giacobini, M., Bersini, H., Bourgine, P., Dorigo, M., and Doursat, R., editors, *Advances in Artificial Life, ECAL 2011*, pages 646–653, Paris, France. MIT Press.

Silva, P. (2010). Institutional approaches to the micro-macro link within human societies. Technical report, Fundação para a Ciência e a Tecnologia.

Stamatakis, M. and Zygourakis, K. (2010). A mathematical and computational approach for integrating the major sources of cell population heterogeneity. *J. Theor. Biol.*, 266:41–61.

Stirling, T., Wischmann, S., and Floreano, D. (2010). Energy-efficient indoor search by swarms of simulated flying robots without global information. *Swarm Intelligence*, 4(2):21–32.

Tarapore, D. and Carneiro, J. (2010). From bio-inspired to institutional-inspired collective robotics: Epigenetic mechanisms controlling collective configurations in simulated Th cell populations. Technical report, Fundação para a Ciência e a Tecnologia.

Tarapore, D. and Christensen, A. (2010). From bio-inspired to institutional-inspired collective robotics: Simulated and real multirobot experiments. Technical report, Fundação para a Ciência e a Tecnologia.

Tummolini, L. and Castelfranchi, C. (2006). The cognitive and behavioral mediation of institutions: Towards an account of institutional actions. *Cognitive Systems Research*, 7(2-3):307–323.

Vuillemin, J. (1978). A data structure for manipulating priority queues. *Communications of the ACM*, 21:309–315.

Waibel, M., Keller, L., and Floreano, D. (2009). Genetic Team Composition and Level of Selection in the Evolution of Cooperation. *IEEE Transactions on Evolutionary Computation*, 13(3):648–660.

Winfield, A. F. T., Liu, W., Nembrini, J., and Martinoli, A. (2008). Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence*, 2(2-4):241–266.

Wolfram Research, I. (2008). *Mathematica Edition: Version 7.0*. Wolfram Research, Inc.

Zykov, V., Mytilinaios, E., Adams, B., and Lipson, H. (2005). Self-reproducing machines. *Nature*, 435:163–164.