Modelling and Optimisation of Adaptive Foraging in Swarm Robotic Systems

Wenguo Liu · Alan FT Winfield

Received: date / Accepted: date

Abstract Understanding the effect of individual parameters on the collective performance of swarm robotic systems in order to design and optimise individual robot behaviours is a significant challenge. This paper presents a macroscopic probabilistic model of adaptive collective foraging in a swarm of robots, where each robot in the swarm is capable of adjusting its time threshold parameters following the rules described in (Liu et al, 2007). The swarm adapts the ratio of foragers to resters (division of labour) in order to maximise the net swarm energy for a given food density. A probabilistic state machine (PFSM) and a number of difference equations are developed to describe collective foraging at a macroscopic level. To model adaptation we introduce the new concepts of the sub-PFSM and private/public time thresholds. The model has been extensively validated with simulation trials, and results show that the model achieves very good accuracy in predicting the group performance of the swarm. Finally, a real-coded genetic algorithm is used to explore the parameter spaces and optimise the parameters of the adaptation algorithm. Although this paper presents a macroscopic probabilistic model for adaptive foraging, we argue that the approach could be applied to any adaptive swarm system in which the heterogeneity of the system is coupled with its time parameters.

Keywords macroscopic probabilistic modelling · swarm robotics · collective foraging

1 Introduction

In recent decades swarm intelligence (SI) has gained increasing attention as a bio-inspired approach to coordinating behaviours of groups of simple robots in multi-robot systems. Case studies include flocking/aggregation (Matarić, 1995; Vaughan et al, 2000; Tanner et al, 2004; Dorigo et al, 2004; Garnier et al, 2008); collective clustering/sorting (Holland and Melhuish, 1999; Martinoli et al, 1999; Wilson et al, 2004); collective searching/inspection (Reif and Wang, 1999; Dudenhoeffer et al, 2001; Correll and Martinoli, 2009); cooperative transport/handling (Kube and Bonabeau, 2000; Ijspeert et al, 2001; Gross and Dorigo, 2004)
and collective foraging (Krieger et al., 2000; Labella et al., 2006). Although there is no central controller governing the behaviour of the swarm and the robots themselves often have very limited sensing, communication and computation, complex collective behaviours emerge from local interactions among the robots and between the robots and the environment with relatively simple individual control rules. Systems based on the principles of SI, known also as swarm robotic systems, emphasise self-organisation and distributedness in a large number of robots. With homogeneity and simplicity as design goals at the individual unit level, the main advantages of the swarm approach lie in the properties of scalability, adaptivity and robustness. However, bottom-up design cannot provide us with quantitative prediction of the swarm performance. In order to design and optimise individual robot behaviours, and hence achieve the desired collective swarm properties, we need to understand the effect of individual parameters on the group performance. Real robot experiments and simulations are the most direct way to observe the behaviour of the system with different parameters. However, trials with real or simulated robots do not scale well as the size of the system grows. It is therefore impractical to search the whole design parameter space to find the best solutions using a trial and error approach.

Mathematical modelling and analysis offers both an alternative and complement to experiments and simulation, and attention has been directed in recent years to addressing the modelling problem in swarm robotics using probabilistic approaches. One such approach is macroscopic modelling, which aims to directly describe the overall collective behaviour of the system. One of the fundamental elements of the macroscopic probabilistic model are the Rate Equations, which have been successfully applied to a wide variety of problems in physics, chemistry, biology and the social sciences. For instance, Sumpter and Pratt (2003) developed a general framework for modelling social insect foraging systems with generalised rate functions (differential equations). Sugawara and coworkers (Sugawara and Sano, 1997; Sugawara et al., 1999) first presented a simple macroscopic model for foraging in a group of communicating and non-communicating robots, with analysis under different conditions. Lerman and Galstyan (2001, 2004) proposed a more generalised and fundamental contribution to macroscopic modelling in multi-agent systems. Lerman et al (2001) developed a macroscopic model of collaborative stick-pulling, and the results of the macroscopic model quantitatively agree with both embodied and microscopic simulations. Lerman (2002) presents a mathematical model of foraging in a homogeneous multi-robot system to understand quantitatively the effects of interference on the performance of the group. Agassounon et al (2004) used the same approach to capture the dynamics of a robot swarm engaged in collective clustering.

Rather than using a time-continuous model, Martinoli and coworkers (Martinoli et al., 2004) considered a more fine-grained macroscopic model of collaborative stick-pulling which takes into account more of the individual robot behaviours, in the discrete time domain, using difference equations. They suggest that time-discrete models are the most appropriate solution for the level of description characterised by logical operators and behavioural states. Similarly, Correll and Martinoli (2005) used a macroscopic probabilistic model for analysis of beaconless and beacon-based strategies for a swarm turbine inspection system, and to find an optimal collaboration policy minimising the time to completion and the overall energy consumption of the swarm in Correll and Martinoli (2006). In Winfield et al (2008) the same macroscopic modelling approach has been applied to study a swarm of wireless networked robots in which the movements of the robots are no longer constrained within a bounded environment.

Despite the success of the above examples, there is little existing work on mathematical analysis of adaptive multi-robot systems in dynamic environments with the notable excep-
tion of the work of (Lerman et al, 2006), which extended the macroscopic probabilistic model to distributed robots that adapt their behaviour based on estimates of the global state of the system. In this work robots engaged in a puck collecting task need to decide whether to pick up red or green pucks based on observed local information. The model must therefore take into account the heterogeneities in the robot population. Lerman, et al, claim that the model can be extended to other systems in which robots use a history of local observations of the environment as a basis for making decisions about future actions.

In our previous work (Liu et al, 2007), we presented a simple algorithm for a group of foraging robots which try to maximise the net swarm energy, through adaptive division of labour. Individual robots each have the same threshold-based controllers and two time thresholds are used to regulate the behaviour of the robots, either foraging or resting. Three adaptation cues (internal, social and environmental), based on local sensing and communications, dynamically vary the time spent foraging or resting. The adaptation algorithm has a number of parameters which are used to adjust the contribution of each cue. Simulation results show the adaptation mechanism is able to guide the swarm towards energy efficiency. However, with a set of parameters selected by hand it is not clear that the swarm has the best performance it can achieve, and there are no obvious guidelines for manually finding the optimal parameters. To address these problems we will develop a macroscopic probabilistic model of adaptive foraging to investigate the effect of the individual parameters of the adaptation mechanisms on the performance of the system. In general, the probabilistic modelling approach is built upon the finite state machine of the individual controller and deals with two types of transitions: one happens when certain external conditions are true (transition probabilities based), the other relies on internal time parameters (threshold based). Unlike the model described in (Lerman et al, 2006), where the adaptation mechanism adjusts the state transition probabilities directly, we will consider a different and complementary case in which the adaptive process tunes two deterministic time parameters of the robot controller using more complex adaptation rules. The main challenge of applying the macroscopic probabilistic modelling framework to adaptive foraging lies in the nested dynamic introduced by the time thresholds. Once the model has been developed and validated, we can then use it to optimise the design of the adaptation algorithm.

This paper is organised as follows: Section 2 introduces the collective foraging scenario and our adaptation algorithm. Section 3 develops the macroscopic probabilistic model of collective adaptive foraging in detail. We first outline the steps of such an approach, then explain the derivation of difference equations for each state in the probabilistic finite state machine. Section 4 validates the model using simulation trials for a range of different experiments. A real-coded genetic algorithm is then used to find the optimal parameters for the adaptation algorithm. Finally section 5 concludes the paper.

2 Adaptive collective foraging

In a collective foraging scenario food is randomly scattered in the environment and each food-item collected will deliver a certain amount of energy to the swarm. In order to maximise net swarm energy for a given density of food-items the swarm needs to be able to find an optimal balance between foraging and resting (division of labour). In our previous work (Liu et al, 2007), we describe a threshold-based controller with two internal time thresholds, which are adjusted up or down according to internal rules, within each robot. Figure 1 shows the finite state machine (FSM) for adaptive foraging. Each state represents a discrete behaviour which, together, comprise a foraging controller. $T_1$ is used to count the
time the robot spends searching and is set when the robot moves out of state resting; $T_2$ is set when the robot moves to state resting and counts the time resting in the nest. The transitions from states randomwalk, scanarena, or movetofood to state homing are triggered whenever searching time $T_1$ reaches its threshold $T_s$; such a transition will reduce the number of foragers which in turn minimises the interference caused by overcrowding. The transition between states resting and leavinghome, is triggered when the robot has rested for long enough, i.e. $T_2 \geq T_r$, will drive the robot back to work to collect more food for the colony, which means increasing the number of foragers in the swarm. Note that to keep the diagram clear, with the exception of state resting, the robot can transition from any state to state avoidance — not shown in Figure 1 — whenever obstacles are detected; after the collision avoidance behaviour is completed the robot will return to its previous state.

The individuals in the swarm use three adaptation cues: internal cues (successful or unsuccessful food retrieval); environmental cues (collision with other robots while searching) and social cues (teammate food retrieval success or failure) to dynamically regulate the two internal thresholds. This adaptation is based on the following rules:

$$
T_i^s(k+1) = T_i^s(k) - \alpha_1 C_i(k) + \beta_1 P_i^s(k) - \gamma_1 P_i^f(k)
$$

$$
T_i^r(k+1) = T_i^r(k) + \alpha_2 C_i(k) - \beta_2 P_i^s(k) + \gamma_2 P_i^f(k) - \eta R_i(k)
$$

where $i$ indicates the ID for each robot, $C_i(k)$ counts the contribution from environmental cues, $P_i^s(k)$ and $P_i^f(k)$ for social cues, and $R_i(k)$ for internal cues. The adjustment factors $\alpha_1$, $\alpha_2$, $\beta_1$, $\beta_2$, $\gamma_1$, $\gamma_2$ and $\eta$, whose values are positive real numbers, are used to moderate the contribution of each corresponding cue. Note that the adjustments take place only when certain state transitions happen (or in state resting). In most cases, the contribution of each cue is zero. $C_i(k)$, $R_i(k)$, $P_i^s(k)$ and $P_i^f(k)$ are defined below.

$$
C_i(k) = \begin{cases} 
1 & \text{state } \text{randomwalk} \rightarrow \text{state avoidance} \\
0 & \text{otherwise} 
\end{cases}
$$

$$
R_i(k) = \begin{cases} 
1 & \text{state } \text{deposit} \rightarrow \text{state resting} \\
-1 & \text{state } \text{homing} \rightarrow \text{state resting} \\
0 & \text{otherwise} 
\end{cases}
$$
\[ P^s_i(k) = \begin{cases} 0 & \text{not in \textit{resting} state} \\ SP^s_i(k) & \text{state \textit{deposit} \rightarrow state \textit{resting}} \\ \sum_{j=1, j\neq i}^{N} \{ R^j(k) | R^j(k) > 0 \} \text{ in \textit{resting} state} \end{cases} \] (5)

\[ P^f_i(k) = \begin{cases} 0 & \text{not in \textit{resting} state} \\ SP^f_i(k) & \text{state \textit{homing} \rightarrow state \textit{resting}} \\ \sum_{j=1, j\neq i}^{N} \{ |R^j(k)| |R^j(k) < 0 \} \text{ in \textit{resting} state} \end{cases} \] (6)

where \( SP^s \) and \( SP^f \) represent social cues (food retrieval success and failure respectively), defined as follows:

\[ SP^s(k+1) = SP^s(k) - \delta + \sum_{i=1}^{N} \{ R^i(k) | R^i(k) > 0 \} \] (7)

\[ SP^f(k+1) = SP^f(k) - \delta + \sum_{i=1}^{N} \{ |R^i(k)| |R^i(k) < 0 \} \] (8)

Attenuation factor \( \delta \) is introduced here to simulate gradual decay rather than instantly disappearing social cues. Note that as the social cues are only accessible for the robots in the nest, two categories of robots will be affected. One group are those already resting in the nest, the other are those ready to move to state \textit{resting} from states \textit{homing} or \textit{deposit}; the former can ‘monitor’ the change of social cues and then adjust time threshold parameters, while the latter will benefit from the gradually decaying cues left by teammates. The two situations for updating \( P^s_i(k) \) and \( P^f_i(k) \) are shown in Equations (5) and (6).

3 A probabilistic model for adaptive foraging

For most behaviour-based robotic systems, although the behaviour of a particular robot at a given time is fully determined, the transitions from one state (behaviour) to another exhibit some probabilistic properties over time within the population of the swarm. The central idea of macroscopic probabilistic modelling is to describe the system as a series of stochastic events and use rate equations to capture the dynamics of these events. A general approach to developing a macroscopic probabilistic model for swarm robotic systems can be summarised as follows:

\textbf{step 1} describe the behaviour of the individual robots of the swarm as a finite state machine (FSM);

\textbf{step 2} transform the FSM into a probabilistic finite state machine (PFSM), describing the swarm at a macroscopic level;

\textbf{step 3} develop a system of rate equations for each state in the PFSM, to describe the changing average number of robots between states at a macroscopic level;

\textbf{step 4} measure the state transition probabilities using experiments with one or two real robots, or estimate them using analytical approaches, and then

\textbf{step 5} solve the system of rate equations.

The PFSM and rate equations form the core of the probabilistic modelling approach. In the PFSM, each state represents the average number of robots in that state, rather than the discrete behaviour of the FSM states. The changes of average number of robots in each state of the PFSM over time can then be described using a set of rate equations, either in
continuous time or discrete time. Clearly the complexity of the model depends very much on
the task itself. Consider our adaptive foraging scenario: the goal of the swarm is to maximise
the net energy of the swarm, but this metric is directly coupled with the number of robots
either Resting or non-Resting. The probabilistic model is capable of capturing the dynamics
of the average transitions of robots between states, thus if we can embed the adaptation rules
into the general probabilistic model, the relationship between the low-level parameters (i.e.
adjustment factors of the adaptation rules) and the group metric (net swarm energy) will be
expressed mathematically.

3.1 The probabilistic finite state machine

The foraging controller FSM for an individual robot shown in Figure 1 can be described as
a PFSM for the whole swarm as shown in Figure 2. To reduce the complexity of the model,
the nine states of the FSM have been simplified to five: states movetohome and Deposit in
the FSM correspond to state Deposit in the PFSM; states leavinghome, randomwalk and
scanarena in the FSM correspond to state Searching; states movetofood and grabfood in the
FSM correspond to state Grabbing, states Resting and Homing in the FSM remain the same.
Note that (with the exception of state Resting), each of these PFSM states also includes
robots in state Avoidance, as in the FSM.

By default, a robot will be in state Searching, denoted S. In each time step, it has prob-
ability $\gamma_f$ to find a food-item thus move to state Grabbing, denoted G, in which state it will
move towards the target food-item until it is close enough to grab it. Once the robot successfully grabs the food-item after \( T_g \) time steps, it will move to state Deposit, \( D \). After the robot deposits the food-item, it will stay in state Resting, denoted \( R \), for \( T_r \) time steps and then return to state Searching. Alternatively, if a robot in state \( S \) fails to find a food-item within the search time \( T_s \), it will move to state Homing, denoted \( H \). The same rule applies to a robot in state \( G \) if its searching time limit is reached, even though the robot is moving towards a food-item. Because of competition among robots if more than one robot catches sight of the same food-item clearly only one of them can actually grab it; a robot in state \( G \) therefore has probability \( \gamma_l \) to lose sight of the food-item because it has been already grabbed by another robot, which in turn causes the robot to return to state Searching.

Table 1: Key to the primary notation used in the PFSM and macroscopic model

<table>
<thead>
<tr>
<th>notation</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_X )</td>
<td>number of robots in state ( X ), ( X \in {S,G,D,H,R} )</td>
</tr>
<tr>
<td>( \Delta X )</td>
<td>total number of robots moving into state ( X )</td>
</tr>
<tr>
<td>( \Gamma_X )</td>
<td>number of robots moving into state Homing from state ( X )</td>
</tr>
<tr>
<td>( M )</td>
<td>number of food-items available in the environment</td>
</tr>
<tr>
<td>( \gamma_l )</td>
<td>probability to lose sight of a food-item</td>
</tr>
<tr>
<td>( \gamma_f )</td>
<td>probability to find a food-item</td>
</tr>
<tr>
<td>( \gamma_r )</td>
<td>probability to collide with other robots</td>
</tr>
<tr>
<td>( T_g )</td>
<td>average grabbing time</td>
</tr>
<tr>
<td>( T_h )</td>
<td>average homing time</td>
</tr>
<tr>
<td>( T_d )</td>
<td>average deposit time</td>
</tr>
<tr>
<td>( T_x^{(y)} )</td>
<td>private time threshold, ( x \in {r,s} ), ( y \in {h,d,s} )</td>
</tr>
<tr>
<td>( T_x )</td>
<td>public time threshold</td>
</tr>
</tbody>
</table>

3.2 Rate equations

Let \( N_S(k), N_G(k), N_D(k), N_H(k) \) and \( N_R(k) \) be the average number of robots in states Searching, Grabbing, Deposit, Homing and Resting respectively, at time step \( k \). As the total number of robots in the swarm must remain constant from one time step to the next, if \( N_0 \) represents the total number of robots in the swarm, then we have

\[
N_0 = N_S(k) + N_R(k) + N_G(k) + N_D(k) + N_H(k) \tag{9}
\]

The changes in the average number of robots in each state can be described with a set of difference equations in the discrete-time domain. Consider \( N_S(k) \) first, the number of robots in state Searching at time step \( k+1 \) can be expressed as

\[
N_S(k+1) = N_S(k) + \gamma_l(k)N_G(k) + \Gamma_h(k+1) - \gamma_f(k)M(k)N_S(k) - \Gamma_r(k+1) \tag{10}
\]

The second term on the right-hand side (RHS) of Equation (10) represents the number of robots losing sight of a food-item while moving towards it (in state \( G \)); the probability of
losing sight of a food-item for one robot, $\gamma(k)$, varies from time to time depending on the number of food-items available and the number of robots competing for food. The third term $\Gamma_R(k + 1)$ is the number of robots moving from state $\text{Resting}$ to state $\text{Searching}$ at time step $k$ and will be explained in section 3.4.2. The remaining terms describe the number of robots moving from state $\text{Searching}$ to other states: $\gamma M(k) N_G(k)$ denotes those transferring to state $\text{Grabbing}$, in which $M(k)$ is the number of food-items available in the environment at time step $k$ and $\gamma$ denotes the probability that one robot finds a food-item while searching in the arena. We define there to be, on average, $p_{\text{new}}$ new food-items growing in the arena each time step (Liu et al, 2007). As a robot can only grab one food-item at a time, the total number of food-items collected by the robots is equivalent to the number of robots transferring to state $\text{Deposit}$ at time step $k$, denoted with $\Delta_D(k)$, thus we have

$$M(k + 1) = M(k) + p_{\text{new}} - \Delta_D(k)$$  \hspace{1cm} (11)

The final entry on the RHS of Equation (10), $\Gamma_G(k + 1)$, represents the number of robots transferring to state $\text{Homing}$ from state $\text{Searching}.$

Similarly, for state $\text{Grabbing}$:

$$N_G(k + 1) = N_G(k) + \gamma M(k) N_G(k) - \gamma N_G(k) - \Delta_D(k + 1) - \Gamma_G(k + 1)$$  \hspace{1cm} (12)

Where $\Gamma_G(k)$ counts those robots failing to grab food-items as they run out of searching time and therefore transfer to state $\text{Homing}.$

For state $\text{Deposit}$:

$$N_D(k + 1) = N_D(k) + \Delta_D(k + 1) - \Delta_D(k - T_d)$$  \hspace{1cm} (13)

For state $\text{Homing}$:

$$N_H(k + 1) = N_H(k) + \Delta_H(k + 1) - \Delta_H(k - T_h)$$  \hspace{1cm} (14)

For state $\text{Resting}$:

$$N_R(k + 1) = N_R(k) + \Delta_D(k - T_d) + \Delta_H(k - T_h) - \Gamma_R(k + 1)$$  \hspace{1cm} (15)

Where

$$\Delta_H(k + 1) = \Gamma_R(k + 1) + \Gamma_G(k + 1)$$  \hspace{1cm} (16)

$$\Delta_D(k + 1) = [\Delta_G(k - T_g) - \Omega_G(k - T_g)] \Lambda_G(k; T_g)$$  \hspace{1cm} (17)

$$\Delta_G(k + 1) = \gamma T M(k) N_G(k)$$  \hspace{1cm} (18)

$\Omega_G(k - T_g)$ in Equation (18) represents that number of robots transferring to state $\text{Grabbing}$ from state $\text{Searching}$ at time step $k - T_g$, whose remaining searching time credit is insufficient to allow those robots to grab the food-item successfully. This group of robots will move to state $\text{Homing}$ once their searching time is up during the next $T_g$ time steps (from $k - T_g$ to $k$). $\Lambda_G(k; T_g)$ denotes the fraction of robots successfully grabbing the food-item at time step $k$ after spending $T_g$ steps moving towards it. It is equivalent to the probability that no transition from state $\text{Grabbing}$ to state $\text{Searching}$ is triggered during the time interval $[k - T_g + 1, k]$, and can be expressed as follows:

$$\Lambda_G(k; T_g) = \prod_{i=k-T_g+1}^{k} [1 - \gamma(i)]$$  \hspace{1cm} (19)
3.3 A sub-PFSM for the “searching-grabbing” task

To solve Equations (10) to (18), we need to derive equations for $\Gamma_S$, $\Gamma_G$, $\Gamma_R$ and $\Omega_G$. As shown in Figure 2, $\Gamma_S$, $\Gamma_G$ and $\Omega_G$ (related to $\Delta_D$) are all related to the searching time threshold $T_s$. There is no straightforward way of writing down these equations explicitly, as in the previous section, because of the nesting of time parameters ($T_s$ and $T_g$). Assume a group of robots transfer to state Searching at time 0, then after $T_g$ steps, some robots may move to state Deposit between time steps $T_g$ and $T_s$, the others then move to state Homing at time step $T_s + 1$. Clearly, within time $T_s$, these robots can only be in the states Searching, Grabbing and Deposit. Once we known how these robots are distributed across these three states over time, the number of robots moving to state Homing, i.e. $\Gamma_S$ and $\Gamma_G$, can be obtained. Based on these considerations we introduce a sub-PFSM which includes two states for task “searching-grabbing”, as shown in Figure 3. The “searching-grabbing” task is clearly a part of the overall foraging PFSM of Figure 2, hence we refer to it as a sub-PFSM.

At each time step one new instance of the sub-PFSM is formed with a different initial number of robots in state Searching, which were transferred from state Resting in the previous time step. The sub-PFSM can be identified by its date of birth (DOB), i.e. the time that it is formed. Because of the searching time threshold the sub-PFSM will no longer exist after these robots transfer to other states some time steps later. During the limited-time lifecycle of a sub-PFSM, its subset of robots could split into states Searching or Grabbing, or move into state Deposit after spending $T_g$ steps in state Grabbing. Thus we can develop rate equations to capture the change in number of robots in the states within the sub-PFSM from one time step to the next.

To avoid confusion with the previous notation of the rate equations, let $N'_S(k; i)$ be the number of robots in state Searching in the sub-PFSM, and $N'_G(k; i)$ for state Grabbing, where $i$ indicates the DOB of the sub-PFSM, and $k$ represents the current time step for the sub-PFSM (as in the full PFSM). A mathematical description for the sub-PFSM of Figure 3 can then be developed as follows

\[
N'_S(k + 1; i) = N'_S(k; i) + \gamma(k)N'_G(k; i) - \gamma_f M(k)N'_S(k; i) \tag{20}
\]

\[
N'_G(k + 1; i) = N'_G(k; i) + \gamma_f M(k)N'_S(k; i) - \gamma_l(k)N'_G(k; i) - \Delta'_G(k - T_g; i)\Lambda_G(k; T_g) \tag{21}
\]
Where \( \Delta_{G}^\prime(k - T_g; i) \Lambda_{G}(k; T_g) \) counts the number of robots that will be successfully transferred to state Deposit. \( \Lambda_{G}(k; T_g) \) is obtained through Equation (19). \( \Delta_{G}^\prime(k; i) \) represents the number of robots moving to state Grabbing (in the sub-PFSM) at time step \( k \), where

\[
\Delta_{G}^\prime(k + 1; i) = \gamma_f M(k) N_{G}^\prime(k; i)
\]

(22)

The initial conditions for the sub-PFSM are \( N_{G}^\prime(i; i) = \Gamma_R(i), N_{G}^\prime(i; i) = 0 \).

Although only one sub-PFSM is formed each time step, each sub-PFSM can have different lifetimes because of the adaptation rules (to be discussed in the next section), hence there could be more than one sub-PFSM coming to the end of its life cycle at time step \( k \). To obtain the number of robots that transfer to state Homing, we need to know which sub-PFSMs expire at time step \( k \). Let \( S(k) \) denote the collection of all the DOBs for all of those sub-PFSMs, then we have

\[
\Gamma_{S}(k) = \sum_{i \in S(k)} N_{G}^\prime(k; i)
\]

(23)

\[
\Gamma_{G}(k) = \sum_{i \in S(k)} N_{G}^\prime(k; i)
\]

(24)

\[
\Omega_{G}(k - T_g) = \sum_{m = k - T_g}^{k} \sum_{i \in S(m)} \Delta_{G}^\prime(k - T_g; i)
\]

(25)

### 3.4 Modelling of adaptation rules

At this stage the PFSM model is almost complete, but requires the description of \( \Gamma_R(k) \) and the newly introduced \( S(k) \). Clearly these two variables have close links with the two time thresholds \( T_s \) and \( T_r \). For a swarm with fixed time thresholds, i.e. no adaptation, deriving equations for \( \Gamma_R(k) \) and \( S(k) \) is straightforward. That is \( S(k) = \{ k - T_s \} \) and \( \Gamma_R(k) = \Delta_{D}(k - T_d - T_r) + \Delta_{H}(k - T_h - T_r) \). However for the case with adaptation, a robot in the swarm may change its time threshold parameters based on internal, social or environmental cues over time, thus \( T_s \) and \( T_r \) are no longer fixed and indeed highly dynamic. Equations (1) and (2) show a linear adjustment operation for the time threshold parameters from each cue. The contribution of each cue is proportional to the number of robots transferring into corresponding states, e.g the contribution of environmental cues equals the average number of robots moving into state Avoidance (not shown in the PFSM but included in state Searching). Although the macroscopic model doesn’t itself take the difference between individual robots into account, the sub-PFSM models a subset of robots in the PFSM. It is therefore practical to introduce exclusive time thresholds into the sub-PFSM and apply the adaptation rules to these robots and their time thresholds. The exclusive time thresholds owned by each instance of the sub-PFSM are called private time thresholds. In consequence two public time thresholds, owned by all robots, are introduced into the model to link many of these private ones. The private time thresholds play the role of deciding when the transition from one state to another is triggered, while the public time thresholds are used to accumulate the contributions from all the adaptation cues which have been applied to the swarm. They affect each other in a bi-directional manner, and are described as follows.

### 3.4.1 Private and public time thresholds

With the adaptation rules, the robots successfully retrieving food-items should, in general, have lower values of \( T_r \) than those which failed. We therefore separate the robots in state
Resting into two groups according to which states they transferred from: either state Homing or Deposit, marked as pseudo-states RH and RD as shown in Figure 4. Two private resting time thresholds $T_r^{(d)}$ and $T_r^{(h)}$, corresponding to the robots that transferred from state Deposit and Homing respectively, are introduced into the model. In the same way as those in the sub-PFSMs, each subset of robots which move to state Resting will have their own copy of private resting time threshold $T_r^{(d)}$ or $T_r^{(h)}$. The transitions from state Resting to Searching are decided by these two private resting time threshold parameters. Three private searching time thresholds, $T_s^{(h)}$, $T_s^{(d)}$ and $T_s^{(s)}$, are introduced for the pseudo-states (RH and RD) and the sub-PFSM. Among these three private searching time thresholds, $T_s^{(h)}$ and $T_s^{(d)}$ are used to track the contribution of social cues when the robots are in state Resting, while $T_s^{(s)}$ is used to track the contribution of environmental cues. The transitions from state Searching and Grabbing to Homing are now determined by $T_s^{(s)}$. Clearly, each of these private time thresholds has a limited lifetime. They are created when a certain fraction of robots move to a specific state and destroyed once those robots move to a new state. During their life cycles, their values will be updated according to the adaptation rules. Like the sub-PFSMs, they can be identified with DOBs. For example, the notation $T_s^{(d)}(k;i)$ is used to represent the values of private resting time threshold at time step $k$, where $i$ indicates the time step that this private resting time threshold is formed (its DOB). Using this notation, $S(k)$ in Equations (23) to (25) can be expressed as

$$S(k) = \{ i | k - i < T_s^{(s)}(k - 1;i) \land k - i > T_s^{(s)}(k;i) \}$$  \hspace{1cm} (26)

where $k - i$ represents the time steps elapsed from the DOB until step $k$.

To obtain $T_R(k)$, let $R_H(k)$ and $R_D(k)$ represent the collection of DOBs for the private resting time thresholds of the pseudo-states RH and RD, which come to the end of their life.
cycles at time step $k$, then
\[
\Gamma_R(k) = \sum_{i \in \mathbb{R}_H(k)} \Delta_H(i - T_h) + \sum_{i \in \mathbb{R}_D(k)} \Delta_D(i - T_d)
\]  
(27)

Clearly, $\mathbb{R}_H(k)$ and $\mathbb{R}_D(k)$ can be expressed similar to $\mathbb{S}(k)$
\[
\mathbb{R}_H(k) = \{ i \mid k - 1 - i < T^{(h)}(k - 1; i) \wedge k - i > T^{(h)}(k; i) \} \]  
(28)
\[
\mathbb{R}_D(k) = \{ i \mid k - 1 - i < T^{(d)}(k - 1; i) \wedge k - i > T^{(d)}(k; i) \} \]  
(29)

As each of the private time thresholds has a limited life cycle and is attached to separate fractions of robots, a public searching time threshold $\hat{T}_s$ and a public resting time threshold $\hat{T}_r$ are introduced to help model the adaptation rules applied on the swarm. Figure 4 depicts the relationship between the private and public time thresholds. Each private time threshold has its own influence domain. The public time thresholds are mainly used to accumulate the contribution of corresponding private time thresholds to the swarm. Although they have no direct influence on the transitions from one state to the another, they decide the initial values of the private time thresholds. To link the private and public time thresholds, two operations, ‘inherit’ and ‘merge’, are defined here. The private time thresholds ‘inherit’ the up-to-date public time threshold when they are formed, and will update (‘merge’ into) the corresponding public time threshold at the end of their lifetime based on certain rules. For resting time thresholds, one pair of ‘inherit’ and ‘merge’ operations is applied. While for searching time thresholds the adjustments first take place on $T^{(s)}_s$ and $T^{(d)}_s$ when robots move into and stay in state Resting, in the same way as for the resting time thresholds, then the adjustments continue on $T^{(s)}_s$ when robots are in the states of the sub-PFSM. Two pairs of ‘inherit’–’merge’ operations are necessary for the adjustments of searching time threshold.

### 3.4.2 Modelling adaptation of time thresholds

To complete the model a mathematical description of the private time thresholds $T^{(s)}_s$, $T^{(d)}_s$ and $T^{(h)}_h$ needs to be developed. Following the considerations above, this section gives a detailed derivation of private resting time thresholds $T^{(d)}_s$ and $T^{(h)}_h$. The private searching time threshold $T^{(s)}_s$ can be obtained using the same approach and is detailed in Liu (2008).

As shown in Equation (2), the adjustment of resting time threshold falls into three categories corresponding to the contribution from internal cues, social cues and environmental cues respectively. The internal cues and social cues are applied whenever the transitions to state Resting occur; then, the social cues continue to play a role while the robots are in state Resting. The environmental cues take effect only for robots in state Searching. To embed these three cues into the model they must be dealt with separately.

#### A. Internal cues & social cues

When the robots move to state Resting at time step $i$ a new copy of private resting threshold, either $T^{(h)}_h$ or $T^{(d)}_d$ according to which state they have transferred from, is created and will be destroyed when that fraction of robots leaves state Resting. As depicted in Figure 4, the mathematical modelling of private resting time thresholds can be summarised in three phases, as follows:

**phase 1) when robots move to state Resting**
As soon as the robots move into state Resting, a private resting time threshold is formed for these robots with the initial value of public resting time threshold. According to Equations (4) to (6), the internal cues will be applied to adjust the private resting time threshold first, followed by the social cues. Following Equation (2), we have

\[ T_r^{(h)}(k + 1; k + 1) = \hat{T}_r(k) - \beta_2 SP_s(k) + \gamma_2 SP_f(k) + \eta \]  
\[ T_r^{(d)}(k + 1; k + 1) = \hat{T}_r(k) - \beta_2 SP_s(k) + \gamma_2 SP_f(k) - \eta \]

where $\beta_1, \beta_2, \gamma_1$ and $\gamma_2$ are the adjustment factors for social cues, and $\eta$ is the adjustment factor for internal cues. $SP_s(k)$ and $SP_f(k)$ represent the social cues of the swarm. The first term in the RHS of Equations (30) and (31) represents the ‘inherit’ operation from the public resting time threshold. The second and third terms in the RHS count the contribution of social cues. The final term then denotes the adjustment of internal cues.

For $SP_f(k)$ and $SP_s(k)$, according to Equations (7) and (8), we have

\[ SP_f(k + 1) = SP_f(k) - \delta + \Delta_H(k - T_h) \]  
\[ SP_s(k + 1) = SP_s(k) - \delta + \Delta_D(k - T_d) \]

where $\delta$ is the attenuation factor, while $\Delta_H(k - T_h)$ and $\Delta_D(k - T_d)$ denote the increased value of social cues respectively, following Equation (4).

**phase 2) when robots are in state Resting**

As shown in the final rows of Equations (5) and (6), social cues continue to play a role in adjusting the time thresholds when robots are in state Resting, thus we have

\[ T_r^{(h)}(k + 1; i) = T_r^{(h)}(k; i) - \beta_2 * \Delta_D(k - T_d) + \gamma_2 * \Delta_H(k - T_h) \]  
\[ T_r^{(d)}(k + 1; i) = T_r^{(d)}(k; i) - \beta_2 * \Delta_D(k - T_d) + \gamma_2 * \Delta_H(k - T_h) \]

**phase 3) when robots move to state Searching**

Once the resting robots move into state Searching, a merge operation will be applied to update the public resting time threshold. At each time step there may be more than one sub-set of resting robots running out of resting time. In order to calculate the contribution that the private resting time thresholds make to the public resting time threshold $\hat{T}_r$, we need to know:

- the number of robots which leave state Resting in the current time step, and
- the impact of social cues and internal cues on the private resting time thresholds $T_r^{(h)}$ and $T_r^{(d)}$ during their lifecycles.

The contribution of each fraction of reactive robots (from state Resting to Searching) to the public resting time threshold can be expressed as the product of the number of robots and the change of the corresponding private resting time threshold. Let $\Delta_{i_r^{(h)}}$ and $\Delta_{i_r^{(d)}}$ be the total contribution provided by the resting robots transferring from state Homing and Deposit respectively, then

\[ \Delta_{i_r^{(h)}}(k) = \sum_{i \in H(i)} \Delta_H(i - T_h)[(T_r^{(h)}(k; i) - \hat{T}_r(i - 1))] \]  
\[ \Delta_{i_r^{(d)}}(k) = \sum_{i \in D(i)} \Delta_D(i - T_d)[(T_r^{(d)}(k; i) - \hat{T}_r(i - 1))] \]
Where $R_D(k)$ and $R_H(k)$ are defined in Equation (28) and (29).

**B. Environmental cues**

The environmental cues adjust the resting time threshold when robots move from state Searching to Avoidance. Although the change of resting time threshold in this case will not affect the behaviour of the robots until they return home, they make contributions to the public resting time threshold $\hat{T}_r$. This contribution can be expressed as $\alpha_2 \gamma_r N_S(k)$, where $\alpha_2$ is the adjustment factor, $\gamma_r$ is the probability that one robot will collide with another robot when searching for food, and $\gamma_r N_S(k)$ is the number of robots moving into state Avoidance at time step $k$.

**C. Updating of $\hat{T}_r$ from all cues**

Combining the effect of all cues, the public resting threshold $T_r$ will be updated as follows

$$\hat{T}_r(k + 1) = \hat{T}_r(k) + \frac{\Delta_{\gamma_f}(k) + \Delta_{\gamma_l}(k) + \alpha_2 \gamma_r N_S(k)}{N_0}$$

(38)

Where $N_0$ is the total number of robots in the swarm.

### 3.5 The swarm net energy consumption

Now that we have described collective adaptive foraging mathematically using the above equations, the metric of the system – the net energy income of the swarm – can be expressed accordingly. Assume each robot will consume $E_r$ units of energy in state Resting, and $E_s$ units of energy in all other states, each time step. Moreover, assume that one collected food-item will deliver the swarm with $E_c$ energy units. If $E(k)$ denotes the net energy income for the swarm at time step $k$, then we have

$$E(k + 1) = E(k) + E_c \Delta_D(k - T_d) - E_r N_R(k) - E_s (N_0 - N_R(k))$$

(39)

Where $E_c \Delta_D(k - T_d)$ denotes the energy collected by the robots, while $E_r N_R(k)$ and $E_s (N_0 - N_R(k))$ represent the energy consumed by the robots, in each time step.

### 4 Results – validation and optimisation

Using geometrically estimated transition probabilities ($\gamma_f$, $\gamma_r$, and $\gamma_l$) and time parameters ($T_g$, $T_h$, and $T_d$), whose derivation is outlined in the Appendix, our mathematical model of adaptive collective foraging can be solved approximately using a computer aided numerical analysis approach. This section presents the validation of the mathematical model and uses the model to optimise the adjustment factors.

#### 4.1 Validation of the model

We use the multi-robot sensor-based simulator Stage (Vaughan, 2008) to validate the model; figure 5 shows a screenshot from the simulation. The home (nest) region is located in the
centre of the arena and indicated with a grey colour. This is where each robot deposits food-items collected and rests before resuming its search. A light source in the nest acts as a homing beacon for robots. Each robot has physical dimensions $0.26m \times 0.26m$ and is equipped with three front mounted light intensity sensors for homing, one at the centre and two on either side, $60^\circ$ from centre. The robot senses it is at home with a floor facing colour sensor. The food-items scattered in the arena are small marked boxes which can be sensed – from a distance – by the robot’s front facing camera. The robot processes the image from its camera in order to determine the relative angle between its current heading to the location of the food-item, then turns, moves forward to the target and, when close enough, grabs it with its gripper. As the robot cannot sense its distance to the target, the grippers are equipped with two beam sensors which are triggered whenever a food-item is directly in-between the two paddles of the gripper. The robot is also equipped with three front mounted infra-red (IR) proximity sensors for detecting collisions with other robots or the arena wall (the food-items cannot be sensed by the IR sensors). The robot’s physical parameters are given in Table 2.

4.1.1 case 1: $P_{new}=0$, no adaptation

We first consider a canonical foraging scenario, widely used in robotics research, in which no adaptation takes place. Initially there are 40 food-items randomly scattered in the arena. The task of the swarm is to collect all food-items and to deposit them in the home region. The time parameters $\tau_s$ and $\tau_r$ are set to 100 seconds and 0 seconds respectively in the simulation. As the time step duration is set to 0.25 seconds, we fix $T_s=400$ and $T_r=0$ in the macroscopic model. All adjustment factors are zero. $P_{new}$ is also zero. All other parameters
Table 2: Robot and Environment parameters for simulation and probabilities estimation

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v$</td>
<td>0.15 m/s</td>
<td>Robot forward velocity</td>
</tr>
<tr>
<td>$w_1$</td>
<td>15°/s</td>
<td>Robot rotation velocity to face food-item</td>
</tr>
<tr>
<td>$w_2$</td>
<td>15°/s</td>
<td>Robot rotation velocity to face home</td>
</tr>
<tr>
<td>$\psi_v$</td>
<td>60°</td>
<td>View angle of camera</td>
</tr>
<tr>
<td>$\psi_b$</td>
<td>95°</td>
<td>Proximity sensor detection angle</td>
</tr>
<tr>
<td>$R_v$</td>
<td>2 m</td>
<td>Camera detection range</td>
</tr>
<tr>
<td>$R_b$</td>
<td>0.4 m</td>
<td>Proximity sensor range</td>
</tr>
<tr>
<td>$R_p$</td>
<td>0.13 m</td>
<td>Robot body radius</td>
</tr>
<tr>
<td>$R_h$</td>
<td>0.5 m</td>
<td>Radius of home region</td>
</tr>
<tr>
<td>$R_{inner}$</td>
<td>0.7 m</td>
<td>Inner boundary radius of food growing area</td>
</tr>
<tr>
<td>$R_{outer}$</td>
<td>3 m</td>
<td>Outer boundary radius of food growing area</td>
</tr>
<tr>
<td>$E_r$</td>
<td>10 units/s</td>
<td>Energy consumed per second in non-Resting</td>
</tr>
<tr>
<td>$E_s$</td>
<td>2000 units</td>
<td>Energy delivered per food-item</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>0.25 s</td>
<td>Time step duration</td>
</tr>
<tr>
<td>$t_l$</td>
<td>2 s</td>
<td>Gripper loading time</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.1</td>
<td>Attenuation factor of social cues</td>
</tr>
</tbody>
</table>

are set to the values given in Table 2. We vary the population of the swarm from 3 to 20 and repeat each simulation run 40 times. In parallel, we run the macroscopic probabilistic model with corresponding parameters and initial conditions.

![Graph](a) 5 robots  
![Graph](b) 10 robots

Fig. 6: Comparison of simulated and modelled instantaneous food-items uncollected in the arena, with different swarm sizes. Initially there are 40 food-items randomly scattered in the arena. The error bars show the standard deviation of 40 simulation runs.

Figure 6 plots the number of uncollected food-items over time with swarm sizes of 5 and 10 robots, and clearly shows good agreement between the results from simulation trials and the macroscopic model. In each case the number of uncollected food-items falls rapidly at the start of the experiment, then slows down gradually with time elapsed. For the different sized swarm, the slope of the curves change in a different way. This also shows a clear difference when the group size is smaller (5 robots) and the uncollected number of food-items drops below 10. As the simplifying assumptions made for estimation of transition probabilities and time parameters may be less satisfied for a relatively small swarm (Liu,
(2008), it is not surprising to see that in some cases the model is less quantitatively accurate (although still qualitatively correct).

Fig. 7: Comparison of simulated and modelled time to collect 80% (a) and 90% (b) of food-items. $M(0) = 40, P_{new} = 0$, $\tau_s = 100s$, $\tau_r = 0$.

Next, we plot the time that the swarm takes to collect 80% of food-items against the size of the swarm, in Figure 7:a. Again we see that the results from simulation match well with those predicted by the corresponding macroscopic model. The results from the model show that: when we increase the size of the swarm, the time to complete the task falls until the size of the swarm reaches 11, then increases gradually with the size of swarm increasing. We observe the same trend in the simulation except the optimal size is 13, as shown in Figure 7:a. However, the difference in completion time between the simulation and the model is small, about 8 seconds. If we increase the collection rate to 90%, i.e. the robots need to collect 36 food-items, the optimal size of the swarm shifts from 11 to 10, but the difference between simulated and predicted minimum completion time is still small (about 10 seconds). Thus we have demonstrated that, for this typical foraging task, the macroscopic probabilistic model can be used to predict both the time for the swarm to complete the task and the optimal swarm size with good accuracy.

4.1.2 case 2: $P_{new}$ nonzero, no adaptation, varying resting time threshold

To validate the model with non-zero growth rates $P_{new}$ and examine whether the optimal time thresholds vary with $P_{new}$, we have run six experiments in which we vary the resting time threshold parameter $\tau_r$ from 0 to 200 seconds (in 40 s steps). Each experiment is repeated 10 times and each simulation lasts for 20000 seconds. Additionally, we change the environmental parameter – growth rate $P_{new}$ – from 0.03 to 0.05 with an interval of 0.005 and repeat the six experiments 10 times each. With the same parameters we also compute the macroscopic model.

Plotting the relationship between total net energy of the swarm (after 20000 seconds) and the resting time threshold parameter $\tau_r$ of individual robots, Figure 8 compares the results from the simulation and the model. As we can see, for the same resting time threshold parameter, increasing the growth rate $P_{new}$, increases the total net energy of the swarm. This is not surprising since the more food-items available, the more energy the swarm can
collect. For the same growth rate, when the rate is below 0.04, the total net energy of the swarm increases with $\tau_r$ increasing. However, for the other three situations the total net energy doesn’t increase monotonically with $\tau_r$ increasing. It reaches a maximal value and then falls, i.e. there is an optimal value of $\tau_r$ in order to achieve maximum net energy for the swarm. Moreover, the maximal total net energy and the corresponding critical value of $\tau_r$ vary with $P_{new}$ changing, as shown in Figure 8:(c)(d)(e). Figure 8 illustrates excellent agreement between the predicted and measured net energy of the swarm for different growth rates.
rates, thus demonstrating that it is more convenient to use the macroscopic probabilistic model to find the optimal value of \( \tau \) than by trial and error with simulation.

4.1.3 case 3: \( P_{\text{new}} \) nonzero, with adaptation

The third set of experiments are designed to validate the model with the full adaptation capability presented in section 2. We choose a set of arbitrarily selected adjustment factors to test the macroscopic model, \( \{ \alpha_1, \alpha_2, \beta_1, \beta_2, \gamma, \eta \} = \{ 5, 10, 20, 40, 20 \} \). With the same set of adjustment factors, we have tested the model with different food growth rates. Figure 9 plots the results from both simulation and the macroscopic probabilistic model, in which the growth rate is varied from 0.03 to 0.05. The error bars represent the standard deviations of data recorded from 10 simulation trials. We see that simulation results fit well to the curves obtained from the macroscopic model, though noting a relatively large gap when the growth rate is set to 0.03 (Figure 9(a)).

4.2 Optimisation of the adaptation algorithm using the genetic algorithm

The challenge we face in optimising the adaptation algorithm is the large solution space for the adjustment factors. These adjustment factors are believed to be correlated with each other, but the relationships between them remain unknown. The simulation of collective foraging using the Stage simulator, running on a 2.6 GHz CPU desktop PC, has a relatively low acceleration ratio over real time. For example, the simulation of a swarm of 8 robots can only achieve an acceleration ratio of about 20, and this value will drop quickly if the swarm size is increased. Since there are no obvious guidelines for manually tuning the parameters of the adaptation algorithm, it is clearly not practical to cover the whole solution space with simulation and a trial and error approach. Given that we have a macroscopic probabilistic model that captures the dynamics of the swarm with reasonable accuracy, the selection of optimal parameters for the adaptation algorithm becomes a multi-parameter optimisation problem with the model in the loop. The objective function can be directly defined as the net energy of the swarm. For each evaluation we need to run the model with the candidate parameters. Although running the model numerically is much faster than the simulation and is not sensitive to swarm size (about 12000 times faster than real time on the same computer), a brute force approach would still be impractical. To reduce the search space to a reasonable size certain constraints must be applied to these parameters. Let us assume all adjustment factors are chosen from a series of bounded positive integer values. Let \( X = [\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma, \eta] \) represent the solution of the optimisation problem, then a search space \( Z \) for \( X \) can be then defined as

\[
Z = \{ X \in \mathbb{Z}^7 | \alpha_{\text{min}} \leq \alpha_1 \leq \alpha_{\text{max}}, \alpha_{\text{min}} \leq \alpha_2 \leq \alpha_{\text{max}}, \ldots, \eta_{\text{min}} \leq \eta \leq \eta_{\text{max}} \} \quad (40)
\]

Any appropriate search technique can be used to find an approximate optimal solution for the adaptation algorithm. Clearly, the genetic algorithm (GA) is a good candidate.

To address this optimisation problem we apply a real-coded GA to search the solution spaces directly (i.e. no encode and decode processes are needed). Here \( X \) represent a chromosome in the population. The fitness of a chromosome is defined as the final net energy of the swarm. Initially, the GA creates 30 individuals in a population for evaluation. Then, in each generation, a population of 15 new chromosomes is generated from the current population through crossover. In addition, a mutation operator is applied to each newly generated
Fig. 9: Comparison of simulated and modelled instantaneous energy of the swarm with adaptive foraging.

offspring with mutation probability of 0.1. Here linear crossover and mutation operators are defined from Chang (2007). The newly generated 15 chromosomes will be added into the current population and the 15 worst individuals are removed in order to return the population to its original size. In order to generate 15 offspring, the commonly used roulette-wheel selection strategy is used to pick up the parent chromosomes from the current population for crossover. Note that any new offspring may or may not be selected into the population, depending on whether its fitness is better than the worst chromosome in the population. Such a replacement strategy means that all the best ranked individuals are brought into the next
generation. Therefore, after some generations, the individuals in the population might all be near-optimal solutions to the original problem.

4.2.1 Rough estimation of the ideal optimal performance

As the best performance of the swarm with adaptation is hard to determine exactly, in order to assess the performance of the proposed steady-state GA, we need to estimate the ‘ideal’ optimal performance of the swarm for given environmental conditions (food density). Clearly, to obtain the maximal net energy, the swarm needs to collect as much energy as possible whilst keeping energy consumption as low as possible. For a growth rate \( P_{\text{new}} \), \( P_{\text{new}} \times T_{\text{dur}} \) food-items grow during a period \( T_{\text{dur}} \). Since – ideally – all of them would be collected during this period, then the total energy retrieved by the robots is

\[
E_{\text{retrieval}} = E_s P_{\text{new}} T_{\text{dur}}
\]  

(41)

The minimum energy consumed by the swarm with \( N_0 \) robots can be expressed as

\[
E_{\text{consumed}} = E_s P_{\text{new}} T_{\text{dur}} T_{\text{ret}} + E_r (N_0 T_{\text{dur}} - P_{\text{new}} T_{\text{dur}} T_{\text{ret}})
\]  

(42)

The first term in the RHS of Equation(42) represents the energy consumed by the robots in order to collect all food-items. The second term represents the energy consumed by the robots resting in the nest. \( T_{\text{ret}} \) is the minimal average time for a successful retrieval, which can be expressed as the sum of \( T_g \) and \( T_d \). Thus, the ideal optimal net energy that the swarm can achieve is expressed as follows.

\[
E_{\text{net}} = E_s P_{\text{new}} T_{\text{dur}} - (E_s - E_r) P_{\text{new}} T_{\text{dur}} (T_g + T_d) - E_r N_0 T_{\text{dur}}
\]  

(43)

4.2.2 Optimal performance

Next we run the steady-state GA to optimise parameter selection under different environmental conditions. We set the population size to 30 individuals and, in each generation of the GA, the worst half will be replaced. The GA records the scores of the best-of-generation individuals for the most recent 30 generations. The termination condition is that the same best-of-generation individual repeats in the last 30 generations. Table 3 gives the constraints applied to the search space. Adjustment factors are now selected from interval-based sets. For example, \( \alpha_1 \) could be 0, 2, 4, 6, . . . 64. The boundary for each parameter is selected intuitively. To evaluate the fitness of each chromosome, the macroscopic model is run with \( T_{\text{dur}} = 20000 \) seconds. Different environmental conditions are considered by choosing \( P_{\text{new}} = 0.040 \) or \( P_{\text{new}} = 0.045 \). Three cases are chosen to test the GA:

- case 1: \( P_{\text{new}} = 0.040 \), both \( T_r \) and \( T_g \) are adjustable;
- case 2: \( P_{\text{new}} = 0.045 \), both \( T_r \) and \( T_g \) are adjustable;

### Table 3: Constraints for the adjustment factors.

<table>
<thead>
<tr>
<th>parameters</th>
<th>( \alpha_1 )</th>
<th>( \alpha_2 )</th>
<th>( \beta_1 )</th>
<th>( \beta_2 )</th>
<th>( \gamma_1 )</th>
<th>( \gamma_2 )</th>
<th>( \eta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>max</td>
<td>64</td>
<td>64</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>interval</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
Figure 10 plots the evolution of average fitness of the population for these three cases. The rapid improvement in the average fitness can clearly be seen in the first few generations in Figure 10. The corresponding best-of-generations are shown in Table 4. As the chromosome in the GA is mapped directly to the solution space for the adaptation algorithm, Table 4 gives also the net energy of the swarm obtained from the macroscopic model (predicted), the simulation, and Equation (43) (ideal). For each case, the predicted net energy of the swarm has reached over 90% of the ideal values (91.2%, 93.5% and 90.6% respectively). Given that 1) $T_{ret}$ is roughly estimated 2) no failure retrievals are taken into account 3) we assume all the food-items are collected during $T_{dur}$ and 4) no interference between robots is taken into account when calculating the ideal net energy of the swarm, we believe that the predicted optimal net energy for the swarm, shown in Table 4 under different environmental conditions (growth rate), is the near-optimal net energy that the swarm can achieve.
We also see in Table 4 that, although there is no adaptation of searching time threshold in case 3, the swarm can still achieve near-optimal performance with a certain combination of adjustment factors, obtained from the GA. This implies that the searching time threshold has less effect on the performance of the swarm than the resting time threshold. Consider the roles of these two time threshold parameters: the resting time threshold determines how long the robots have to rest in the nest, with a direct effect on the energy of the swarm; while the searching time threshold only has an effect when robots take too long searching in a relatively low food density environment. In most cases, however, the robots succeed in finding food before their searching time expires. Thus, adaptation of the resting time threshold parameter is more important than adaptation of searching time threshold.

5 Conclusions

This paper has described a macroscopic probabilistic model developed to investigate the effect of individual parameters on the group performance, and hence optimise the controller design, for a swarm of robots engaged in collective adaptive foraging for energy. A probabilistic finite state machine has been presented, and a set of difference equations derived, to describe collective foraging at the macroscopic level. The first challenge in modelling adaptive collective foraging is that of dealing with nested time parameters and their different priorities. Although it is fairly straightforward to write down most of the difference equations for the proposed PFSM, deriving a mathematical description which captures the changes in number of unsuccessful robots (i.e. that failed to find food), as a function of searching time threshold, is challenging. We have achieved this by introducing the idea of a time-limited sub-PFSM into the model. This sub-PFSM evolves with the full PFSM but represents only the sub-set of robots engaged in the so-called “searching-grabbing” task.

As macroscopic probabilistic modelling approaches are built upon the theory of stochastic processes, they do not take the exact trajectory of individual robots into account. It is therefore difficult to deal with the heterogeneities in the swarm due to the differences in control parameters. With adaptation, the resting time and searching time thresholds are different from robot to robot, even from one time step to the next. The second challenge is, therefore, that of how to build these differences into the macroscopic model. To solve this problem, in conjunction with the idea of the sub-PFSM, we have introduced private resting and searching time thresholds, and their counterparts – public resting and searching time thresholds. The private time thresholds are valid for a sub-set of robots within the sub-PFSM and play the role of deciding when the transition from one state to another is triggered, while the public time thresholds are used to link all of the private time parameters created each time step; they affect each other in a bi-directional manner. The adaptation rules are then modelled by adjusting the private/public time threshold parameters accordingly.

We have validated the macroscopic model using sensor-based simulation. As there are a number of design parameters in our model, different simulation trials have been carried out to test the model by varying the environmental conditions (food growth rate) and adaptation parameters. The results from the macroscopic model match those from simulation with good quantitative accuracy. We furthermore used a real-coded genetic algorithm to help to find a near-optimal solution for the adjustment factors which are used in the adaptation algorithm. Although it is very hard to exactly determine the best performance of an adaptive system due to its dynamics, we have proposed a simple way to estimate the ideal-optimal performance for the swarm under different environmental conditions. The near-optimal performance obtained via the genetic algorithm is very close to the ideal-optimal performance. Moreover,
we have tested the near-optimal solutions in simulation trials, and results again show that there is good quantitative agreement with the macroscopic probabilistic model.

We have also shown that our macroscopic model of adaptive foraging can be used to investigate the efficiency of the system, in terms of swarm size, in the standard robot foraging task, similar to that presented in Lerman (2002). But compared with Lerman’s model, our downgraded model (with $p_{new} = 0$, and no adaption) represents the foraging task with more detailed behaviours, in particular competition between robots, and with no free parameters at all. This is one of the reasons the full model is presented in a fairly complex form. However, the complexity of the model lies mainly in the challenges of modelling the adaptivity of the system from individual robot level to group level; to the best of our knowledge this has not been fully investigated to date. Our model has aimed to represent the adaption rules for the individual robot while minimising complexity. Indeed, we would argue that our PFSM model is as simple as it can be given the purpose of the model, i.e. as a tool to analyse and hence optimise the design of the adaptation rules of the swarm system. We also note that mean-field approaches such as the proposed macroscopic model do not capture the randomness of the system. If that is the main interest of the modelling effort a microscopic probabilistic approach, as introduced by Martinoli et al (1999), should be applied. However, given the complexity of the adaptation rules a microscopic model of adaptive foraging – even if it could be constructed – would also have high complexity.

In conclusion, we have demonstrated that the macroscopic probabilistic modelling approach can be successfully extended to model the heterogeneities of the swarm using sub-PFSMs and averaging techniques. The sub-PFSM solves the nested dynamics problem of the two nested time parameters, and provides a way to model the differences between individuals at a macroscopic level. Although the techniques presented in this paper have been developed to model adaptive foraging, the structure of the PFSM presented for this case study is not atypical, thus we argue that these techniques are generally applicable to the class of heterogeneous (in control parameters) swarm systems with similar dynamics. In swarm robotics macroscopic modelling provides us with a powerful tool for gaining a deeper understanding of the effect of individual robot characteristics on the overall performance of the collective, and therefore guiding performance optimisation of the individual robot controllers.

Acknowledgments

This work was partially supported by the EU Asia-link Selection Training and Assessment of Future Faculty (STAFF) project. We are also indebted to the anonymous reviewers for the many insightful comments which have guided the improvement of this paper.

Appendix: Estimation of transition probabilities and time parameters

Transition probabilities $\gamma_f, \gamma_r, \gamma_l$, and time parameters $T_g, T_h, T_d$, can be obtained analytically by considering robot control strategies together with the following three assumptions:

- food-items are uniformly dispersed in the arena, over time;
- robots are equally likely to occupy any given position in the arena, and
- the relative heading between two robots varies uniformly from $0^\circ$ to $360^\circ$. 
A detailed derivation of these parameters can be found in Liu (2008), but the essential results are summarised as follows. Note that any variables appearing in the following equations but not explained here are design parameters shown in Table 2.

A. Probability to find a food-item $\gamma_f$

$\gamma_f$ can be expressed as the ratio of the detection region scanned by the robot in each time step ($\Delta t$) to the size of the area in which food-items are located.

$$\gamma_f = \frac{\psi_v R_v V}{\pi (R_{outer}^2 - R_{inner}^2)} \quad \text{(44)}$$

B. Probability to collide with teammate $\gamma_r$

The probability of collision with a teammate while in the Searching state changes with the number of active robots, $N_{active}(k) = N_0 - N_R(k)$. If $P_{in}$ is the probability that one teammate robot is in the potential collision area and $P_a$ the probability of that teammate robot triggering the collision sensor, then

$$\gamma_r(k+1) = 1 - (1 - P_{in}P_a)^{N_{active}(k)-1} \quad \text{(45)}$$

Generally, $P_{in}P_a \ll 1$, so the above equation can be simplified to

$$\gamma_r(k+1) = (N_{active}(k) - 1)P_{in}P_a \quad \text{(46)}$$

$P_{in}$ is given by the ratio of the collision area to the search area

$$P_{in} = \frac{2V\Delta t \psi_v (R_b + R_p)}{\pi (R_{outer}^2 - R_{inner}^2)} \quad \text{(47)}$$

and $P_a = 0.5$.

C. Probability to lose sight of the target $\gamma_l$

The probability of losing sight of the target can be expressed as the probability that the robot is not the closest robot to the target food-item when there is at least one other robot competing for that food-item. It is a function of the number of robots competing for the food-item, $N_{fa}$, and the number of food-items one robot may detect at the same time, $M_{fa}$.

$$\gamma_l(k) = 2 \left( 1 - \frac{1}{N_{fa}} \right) \left( 1 - (1 - \frac{p_g}{M_{fa}})^{N_{fa}-1} \right) / T_g \quad \text{(48)}$$

where $p_g = \psi_v/2\pi$ gives the probability that a robot within the detection range will catch sight of the food-item. $N_{fa}$ can be obtained as:

$$N_{fa} = \frac{R_v^2}{(R_{outer}^2 - R_{inner}^2)} (N_S(k) + N_G(k)) \quad \text{(49)}$$

and

$$M_{fa} = \frac{\psi_v R_v^2}{2\pi (R_{outer}^2 - R_{inner}^2)} M(k) \quad \text{(50)}$$

D. Average grabbing time $T_g$
The average time a robot stays in state *Grabbing* can be expressed as the sum of the average times spent on ‘turning to face the food-item’, ‘moving towards the food-item’, and ‘loading the food-item using gripper’, \( t_i \), that is,

\[
\tau_g = \frac{\psi}{4w_1} + \frac{R_i}{V} + t_i \tag{51}
\]

\( \tau_g \) can be discretised to \( T_g \) by dividing by \( \Delta t \).

**E. Average deposit time \( T_d \) and homing time \( T_h \)**

As a robot uses the same strategies to move home in states *Deposit* and *Homing*, we simply assume \( T_h = T_d \). Taking \( \tau_d \) (the continuous time counterpart of \( T_d \)) as an example, it is the sum of the average time a robot spends turning to face home and the average time it spends moving home, then we have

\[
\tau_d = \frac{\pi}{2w_2} + \frac{D_d}{V} \tag{52}
\]

where \( \pi/2 \) is the average angle the robot needs to turn to face home, and \( D_d \) is the average distance the robot then travels. \( D_d \) depends on the distribution of food-items in the arena and is geometrically estimated as

\[
D_d = \frac{2(R_{outer}^3 - R_{inner}^3)}{3(R_{outer}^2 - R_{inner}^2)} - R_h \tag{53}
\]

**References**