Formal Verification of Probabilistic Swarm Behaviours

Savas Konur, Clare Dixon and Michael Fisher

Department of Computer Science, University of Liverpool, Liverpool, UK
{Konur, CLDixon, MFisher}@liverpool.ac.uk

Abstract. Robot swarms provide a way for a number of simple robots to work together to carry out a task. While swarms have been found to be adaptable, fault-tolerant and widely applicable, designing individual robot algorithms so as to ensure effective and correct swarm behaviour is very difficult. In order to assess swarm effectiveness, either experiments with real robots or computational simulations of the swarm are usually carried out. However, neither of these involve a deep analysis of *all* possible behaviours. In this paper we will utilise automated *formal verification* techniques, involving an exhaustive mathematical analysis, in order to assess whether our swarms will indeed behave as required.

1 Introduction

A *robot swarm* is a collection of simple, and often identical, robots which will work together to achieve some task [1, 2, 12]. Whilst the behaviour of each individual robot is fairly easy to understand, it is considerably harder to predict and guarantee the emergent behaviours of the overall swarm. Consequently, it is very difficult to design an individual robot control procedure that, when replicated across all the robots, will guarantee the required swarm behaviour. In this paper we will explore how such robot algorithms can be analysed in a more formal way than simply by simulation or testing.

To exhibit our approach, we have chosen a scenario for which swarm algorithms have already been designed, implemented and tested. Thus, we focus on *foraging robots*, specifically those developed in [10, 11]. The idea is that robots have to search for, and retrieve, food items, bring them back to the '*nest*' and then rest. There are a number of parameters involved here, for example the time the robots spend resting, the probability of any robot finding food, the energy expended in searching, the energy gained by food, etc. In analysing swarm algorithms of this sort, it is particularly important to see how such parameters affect the swarm behaviours in terms of, for example, *overall swarm energy* or the ratio of *searchers* to *resters* within the swarm. By such an analysis we can explore under what conditions the swarm exhibits optimal behaviour.

Swarm analysis is usually carried out either by testing real robot implementations, or by computational simulations; see, for example, [7, 11]. However, each of these only examines a relatively small number of the possible swarm behaviours and so, especially where swarms are to be deployed in safety (or business) critical areas, these approaches guarantee very little about actual swarm behaviours. A general alternative to simulation and testing is to use *formal verification*, and particularly the technique called *model-checking* [3]. Here a mathematical model of *all* the possible behaviours of the system is constructed and then all executions through this model are assessed against a logical formula representing a required property of the system. If there is any possible behaviour that violates the required property, then this is highlighted.

While model checking techniques usually assess a *temporal* property, we will use something more sophisticated. Since there is inherent uncertainty within swarms, here we use a *probabilistic* model checker called PRISM [6]. Thus, we use a *probabilistic model* of the individual robots [10] and so we can potentially analyse not just the temporal, but also the probabilistic, properties of the swarm. So, our aim is to take an existing robot swarm algorithm, model it in our framework, and then automatically analyse all possible runs through the system via the PRISM model checker. While we can verify properties of individual robots, we are particularly concerned with *global* swarm behaviour. Thus, we will show that we can not only re-create the simulation results from relevant papers [10], but can also show that certain properties hold of *all* possible runs/configurations, i.e. we can *formally verify* swarm behaviour. This extends beyond testing and simulation of robot swarms to formal verification via probabilistic model checking and provides the potential for much deeper analysis of swarm behaviours.

In Section 2 we introduce the foraging robot scenario and in Section 3 we explain how it is to be modelled and verified. In Section 4 we present several experiments based on this model. These are carried out using the PRISM model checker and comprise both simulation and verification activities. In Section 5, we provide concluding remarks.

2 The Foraging Robot Scenario

The foraging robot scenario we consider follows that presented in [10]. Within a fixed size arena, there are a number of foraging robots, i.e. they must search a finite area and bring food items back to the nest. Food is placed randomly over the arena and more may appear over time. The food items collected will increase the energy of swarm, but searching for food items will use up energy and there is no guarantee that robots will actually *find* any food. The behaviour of each robot in the system is represented by the probabilistic state machine in Fig. 1, with states;

- SEARCHING, where the robot is searching for food items;
- GRABBING, where the robot attempts to grab a food item it has found;
- DEPOSITING, where the robot moves home with the food item;
- HOMING, where the robot moves home without having found food;
- RESTING, where the robot rests for a particular time interval.

Associated with these states are both time-out conditions and probability values:

T_s: the maximum amount a time a robot can continue searching;
T_g: the maximum amount of time a robot can attempt grabbing;
T_d: (T_h/T_r) the average time spent depositing (homing/resting);
γ_f: the probability of finding a food item;
γ_g: the probability of grabbing a food item; and
γ_l: the probability of 'losing' sight of a food item, e.g. due to robot interference.

The probabilistic finite state machine in Fig. 1 is adapted from [10], the main difference being that we ignore avoidance in this initial investigation. All robots are initially in the state SEARCHING. In each time step, robots move to the GRABBING state with probability γ_f (the chance of a robot finding food). Robots stay in the SEARCHING

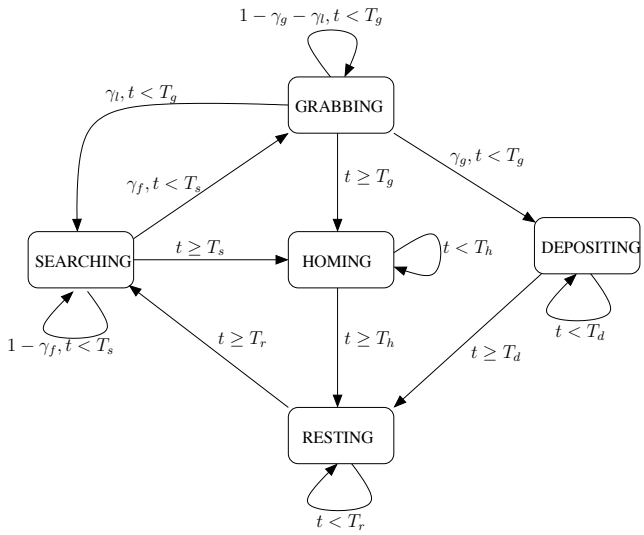

Fig. 1. Probabilistic Finite State Machine for a Single Foraging Robot [10].

state with probability $1-\gamma_f$. If a robot cannot find food within T_s time steps, it will move to the HOMING state. In the GRABBING state the robots move to the DEPOSITING state with probability γ_g (the chance of grabbing the food), move to the SEARCHING state with probability γ_l (the chance of losing sight of food as it has been grabbed by another robot), and stay in the GRABBING state with probability $(1-\gamma_g)-\gamma_l$. If a robot cannot grab a food item in T_g time steps, it will move to the HOMING state. The robots in the HOMING (respectively DEPOSITING, RESTING) state take T_h (respectively T_d, T_r) time steps to return back to the nest (respectively deposit food, rest) before they move to the next state.

3 Modelling and Verifying the Scenario

The essence of *formal verification* is to analyse a logical requirement (typically within *formal logic*) against all possible behaviours of the system in question. The fastest, and most widely used, approach is called *model-checking* [3]. Here, a structure (such as a finite-state automaton) describing all possible system behaviours is exhaustively (and automatically) checked against the required logical properties. Thus, given that the state-machine in Fig. 1 is a *probabilistic* model, we can analyse both *probabilistic* and *temporal* properties of such robots using a *probabilistic model checker*, such as PRISM [6]. Indeed, when we verify properties of an individual probabilistic state-machine, our input to PRISM is a probabilistic model (technically a *discrete-time Markov Chain*) and a property which can be represented in a number of *probabilistic temporal logics*, such as PCTL [5]. PCTL can be used to represent quantities such as "the probability a robot eventually reaches the nest", "the probability that the energy in the system

is greater than E", etc., as well as standard temporal properties. Using PRISM, we can also compute the minimum or maximum probability over a range of possible configurations or parameters of a model, producing a form of best/worst-case analysis.

However, in this paper we are particularly concerned with verifying the properties of swarms comprising large numbers of robots. Although PRISM is generally quite efficient, allowing us to analyse models with as many as 10^{10} states (see, for example, [4]), a naive application of PRISM to robot swarm verification may generate many more states. For example, if we built a product state-machine from multiple copies of the Fig. 1 state-machine then the size of the resulting model would be *huge*. So, rather than representing each robot as a different probabilistic state machine and then taking the *product* of all these machines to generate the whole system, we use a *counting abstraction* approach. This is particularly useful if there are many identical, independent processes, as is the case in a robot swarm, and allows us to abstract away from low-level probabilistic details and so just consider global population behaviour.

The basic idea is as follows. Since we know that all the robots are modelled by identical probabilistic state machines, then we actually model the whole system by *one* state machine with exactly the SEARCHING, HOMING, etc, states we saw in Fig. 1. However, to each of these states we add a counter which is used to record how many robots are *actually* in that state at that moment. Thus, if 20 robots are searching then the counter in the SEARCHING state will be 20. By examining Fig. 1 we can work out how many of these should move to GRABBING, how many should move to HOMING, and how many should remain in SEARCHING at each step. Thus, in addition to the 5 states, each state is labelled with a set of difference equations explaining how the numbers of robots associated with each state evolve. It is important to note that we are abstracting away from local probabilities and now considering a more global view.

Due to lack of space we do not show the (difference) equations defining how the numbers of robots in each of the 5 key states changes over time. But we remark that fractional values of numbers of robots are rounded to the nearest whole number since we work with a discrete state space.

4 Experiments

In this section, we present the results from running PRISM on our model with different properties and parameters. We run them both in *simulation mode*, whereby we generate a random run, and in *verification mode*, whereby we assess all possible runs against a PCTL formula. The properties we check use PCTL syntax, which comprises the usual operators from classical logic such as $\wedge$ (and), $\vee$ (or) and $\Rightarrow$ (implies), as well as probabilities for example $P^{=1}$ (i.e. with probability 1) and temporal operators such as $\Box\varphi$ (φ holds at all present/future moments) $\diamondsuit\varphi$ (φ holds at some present/future moment), $\varphi\,\mathcal{U}\,\psi$ (φ holds until ψ holds).

Energy Calculation. Before discussing the experiments, we will explain how the swarm energy is calculated. In a swarm, each robot consumes a certain amount of energy at each time step. We assume that a robot consumes E_s, E_g, E_r and E_h units of energy at each step in SEARCHING, GRABBING, RESTING and HOMING, respectively, and each

food-item delivers the swarm E_d units of energy (we assume that E_d is net energy, i.e. it is the energy obtained from the food carried minus the energy consumed in the depositing state; we also assume that a robot can carry only one food-item.) The total swarm energy in the next time instance, denoted by $En(t+1)$, is calculated as follows:

$$En(t+1) = En(t) + E_d N_d^{T_d}(t) - E_s N_s(t) - E_g N_g(t) - E_r N_r(t) - E_h N_h(t)$$

where $N_d^{T_d}(t)$ denotes the number of robots that have been in DEPOSITING for T_d time steps; $N_s(t), N_g(t), N_r(t)$ and $N_h(t)$ denotes the number of robots that are in the SEARCHING, GRABBING, RESTING and HOMING states, respectively, at time t. It is important to note that in order to ensure that we have finitely many states we discretise the energy values, and model En as discrete state variable.

4.1 Swarm Model with Resting Timeout

We will start our experiments with a model based on the state structure in Fig. 1 and then will gradually enhance this to be more sophisticated. So, we begin with experiments based on the basic model in which robots leave the RESTING state after waiting for T_r time steps. The energy parameters used are as follows: $E_d = 250$, $E_r = 2$, $E_h = 4$, $E_s = E_g = 50$, and $En(0) = 2000 \times 10^3$ (the initial swarm energy at $t = 0$). We also take $T_s = T_g = T_h = T_d = T_r = 50$ seconds, and $\gamma_l = 0.1$, $\gamma_g = 0.8$. In the sequel, we take N as the total number of robots. Fig. 2 compares the total energy of a swarm of 100 robots with the different (but constant) γ_f (probability of finding food) values. As seen in Fig. 2, if the probability of finding food is below 0.5, i.e. less food is available for the swarm, then the total energy of the swarm decreases. If the probability of finding food is greater than 0.5, i.e. more food is available, then the swarm gains energy.

In PRISM, we can test various runs (i.e. performing simulations as in Fig. 2); but this does not guarantee that a property is true in *all* cases. Using the verification module of PRISM we can also *verify* whether a property holds for all possible runs; or we can determine the actual probability of a property being true. Assume we want to determine the probability that "for an arbitrary number of robots and food finding probability the swarm energy is equal to or greater than the initial energy from a time point t_A". This implies that from a time point t_A the total energy of the swarm never goes below the initial energy. We assume γ_f takes discrete steps with increments of 0.01 in the range $[0, .., 1]$, $N = [100, .., 500]$, $t_A = 2000$ and $t_{max} = 10000$. We can specify this property in PCTL as follows:

$$P^{=?}((t < t_A \vee En \geq E(0))\, \mathcal{U}\, t = t_{max})$$

We verified this formula using PRISM, and the probability returned is 0.59. This result validates the simulation shown in Fig. 2. In Fig. 2, γ_f values remained constant throughout the experiment. In Fig. 3 we use a variable γ_f value that decreases over time, i.e. modelling the situation when food gradually becomes more scarce. So, γ_f is 1.0 at $t = 0$, and reduces by 0.1 in every 1000 seconds. Running some simulations, we see that the total swarm energy initially increases, since the probability of finding food is high, i.e. there is much food available for the swarm. When the probability of finding

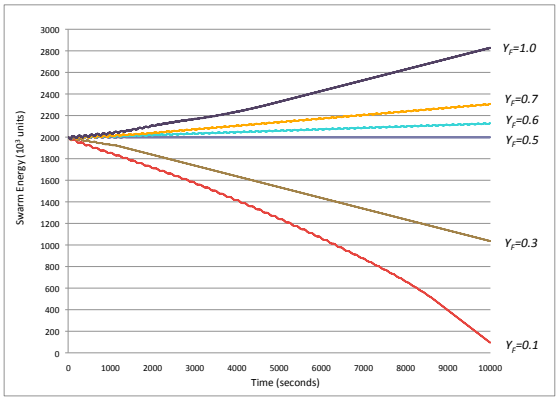

Fig. 2. Total swarm energy vs. the (constant) probability of finding food (γ_f).

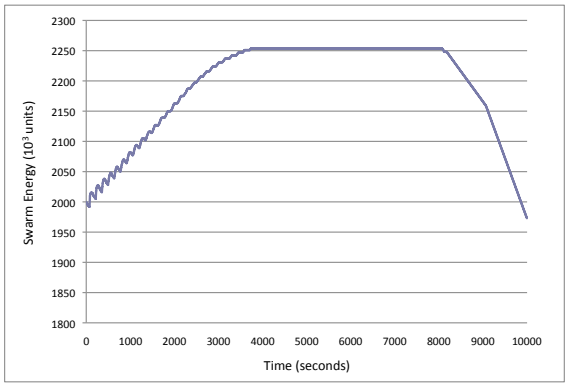

Fig. 3. Total swarm energy vs. the (decreasing) probability of finding food (γ_f).

food decreases, i.e. food availability reduces, the energy gain becomes equal to the energy spent by the swarm. After a while the energy gained becomes less than the energy spent, since the food becomes scarce, and therefore the total swarm energy decreases.

Again, rather than just running individual experiments, we can *verify* some properties of all executions. For example, the following property states that "the total energy never reduces below a certain value E (before timeout)", specified in PCTL as $P^{=1}\,\Box(Energy > E)$. We verified this formula in PRISM (We assumed $E = 1900 \times 10^3$ and $N = [100, .., 500]$). This property can be useful to determine a set of robots can achieve an important task within a certain period, since we can be sure they will always have energy above a certain threshold value. We also queried the following formula in PRISM

$$P^{=?}(true\ \mathcal{U}\,(En > E \wedge P^{=1}\diamondsuit(En > E \wedge t = t_{max})))$$

assessing the probability that if the total energy exceeds E at some time, then the energy level in the end will be above E. PRISM returned a probability value of 0.31 for $E =$

3000×10^3, $N = [100, .., 500]$ and $t_{max} = 10000$. This is an expected result because food becomes scarce as time passes, and so it is likely that the energy level decreases.

4.2 Swarm Model without Resting Timeout

In Fig. 1, we assumed that if a robot moves to the RESTING state from HOMING or DEPOSITING, it waits T_r time steps in RESTING. We now change this scenario and assume that the robots do not wait in the RESTING state for a fixed amount of time before moving to SEARCHING, but wait there depending on a probability γ_s. This probability, in turn, depends on the number of robots in the DEPOSITING and HOMING states. Namely, robots move from RESTING to SEARCHING states with probability

$$\gamma_s = \frac{\lambda N_d^{T_d}(t) - N_h^{T_h}(t)}{N},$$

and stay in the RESTING state with probability $1 - \gamma_s$. In the above, λ is the *energy gaining parameter*, and $N_d^{T_d}(t)$ (respectively $N_h^{T_h}(t)$) denotes the number of robots that have been in DEPOSITING (respectively HOMING) for T_d (respectively T_h) time steps. Intuitively, we can think of this new scenario as follows: since each robot brings a food item in DEPOSITING state, if more robots move to the DEPOSITING state and less robots move to the HOMING state, then more robots will move to the SEARCHING state. As can be seen, γ_s is proportional to the *energy gaining parameter* λ. That is, if we increase the value of λ, then more robots will move to the SEARCHING state.

Using verification we can also establish several properties. For example, we have checked the PCTL property $P^{=?}\,\Box((t > t_A) \Rightarrow\ (N_s \geq n))$, specifying the probability that the number of searching robots will never reduce below n after t_A seconds. The probability that PRISM returned is 0.99 for $t_A = 2000$ and $n = 10$. We also assumed λ takes discrete steps with increments of 0.01 in the range $[0, .., 1.5]$ and $N = [100, .., 200]$. This result shows that the number of searching robots is never below n for *almost* all time points from t_A.

In addition to the scenarios above, we also considered a scenario where γ_g depends on the number of robots searching. Thus, if a robot is in the GRABBING state, then it can grab a food with a probability $\gamma_g = \alpha/N_s$, where α is a constant and N_s is number of robots searching. Due to lack of space, we omit the formal analysis of this scenario.

5 Conclusions

In this paper we have taken a probabilistic state transition system for foraging swarm robots from [10] and used it as the basis for verification of global swarm behaviour using the PRISM model-checker. Rather than instantiating such a transition system for each robot and performing local verification, we adopt a macroscopic approach and represent the whole swarm using one transition system calculating the number of robots in each state based on a combination of the number of robots in the previous state and the probability that robots change state. This allows us to simulate the global foraging robot scenario for a number of parameters. In particular we investigate the changes to swarm energy relating to changing the probability of finding food and differing resting

timeouts. We also experimented with variable probabilities including the probability of moving from resting to searching and the probability of grabbing.

It is important to note that the use of this counting abstraction allows us to analyse the behaviour of large numbers of robots — otherwise, we would not have been able to carry out local verification even with tens of robots composed together. Using this approach we can formally verify that certain behaviours will *always* happen. This is not something that can be done easily with either simulation or testing.

Foraging robots have been studied in a number of other papers, for example [8, 7, 9–11]. The foraging robot scenario we use here is from [10] however as stated previously the main difference being that we ignore avoidance in this paper.

Extensions of this work include extending the work we present here to incorporate avoidance so we can compare with work such as [8], incorporating further robot interactions, and developing PRISM models for other domains such as "stick pulling".

Acknowledgements.

The authors would like to acknowledge the helpful comments from the anonymous reviewers. This work was partially supported by EPSRC research project EP/F033567.

References

1. Beni, G.: From Swarm Intelligence to Swarm Robotics. In: Proc. International Workshop on Swarm Robotics (SAB). LNCS, vol. 3342, pp. 1–9. Springer (2005)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. J. Artificial Societies and Social Simulation 4(1) (2001)
3. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (1999)
4. Duflot, M., Kwiatkowska, M., Norman, G., Parker, D.: A Formal Analysis of Bluetooth Device Discovery. Int. J. Software Tools Technology Transfer 8(6), 621–632 (2006)
5. Hansson, H., Jonsson, B.: A Logic for Reasoning about Time and Reliability. Formal Aspects of Computing 6, 102–111 (1994)
6. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A Tool for Automatic Verification of Probabilistic Systems. In: Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 3920, pp. 441–444. Springer (2006)
7. Labella, T.H., Dorigo, M., Deneubourg, J.L.: Efficiency and Task Allocation in Prey Retrieval. In: Biologically Inspired Approaches to Advanced Information Technology. LNCS, vol. 3141, pp. 274–289. Springer (2004)
8. Lerman, K., Galstyan, A.: Mathematical Model of Foraging in a Group of Robots: Effect of Interference. Autonomous Robots 13(2), 127–141 (2002)
9. Lerman, K., Martinoli, A., Galstyan, A.: A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems. In: Proc. International Workshop on Swarm Robotics (SAB). LNCS, vol. 3342, pp. 143–152. Springer (2005)
10. Liu, W., Winfield, A., Sa, J.: Modelling Swarm Robotic Systems: A Study in Collective Foraging. In: Proc. Towards Autonomous Robotic Systems (TAROS). pp. 25–32 (2007)
11. Liu, W., Winfield, A., Sa, J., Chen, J., Dou, L.: Strategies for Energy Optimisation in a Swarm of Foraging Robots. In: Proc. 2nd International Workshop on Swarm Robotics (SAB). LNCS, vol. 4433, pp. 14–26. Springer (2007)
12. Sahin, E., Winfield, A.F.T.: Special Issue on Swarm Robotics. Swarm Intelligence 2(2-4), 69–72 (2008)