Verifying Properties of Robot Swarms

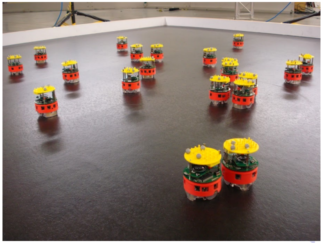
Clare Dixon
Dept. of Computer Science
University of Liverpool, UK
cldixon@liverpool.ac.uk

in collaboration with
Michael Fisher, Savas Konur, M. Carmen Fernandez-Gago,
Paul Gainer, Chengxiu Zeng (Liverpool)
Wenguo Liu, Jin Sa, Alan Winfield (Bristol Robotics Lab)

[Images from BRL]

Introduction

- A robot swarm is a collection of simple (often identical) robots working together to carry out some task.
- Each robot has a relatively small set of behaviours and is typically able to interact with other (nearby) robots and with its environment.
- The use of robot swarms has become increasing appealing in areas which are hostile to humans such as underwater environments, contaminated areas, or space.

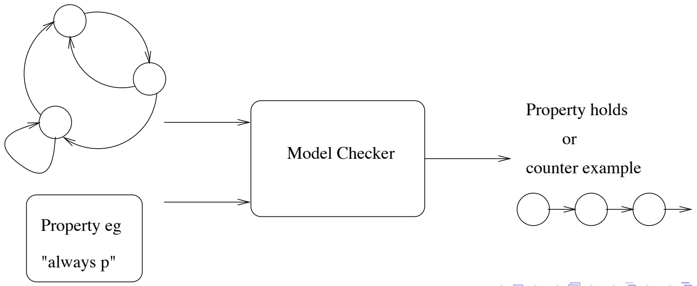


Specifying and Verifying Robot Swarms

- Swarms are thought to be more fault tolerant and more cost effective than one or two highly complex robots.
- However, it is challenging for designers to formulate individual robot behaviours so that the emergent behaviour of the swarm as a whole is guaranteed to achieve the task of the swarm.
- The analysis of swarm behaviour is typically carried out by experimenting with real robot swarms or by simulating robot swarms and testing various scenarios.
- In both these cases any errors found will only be relevant to the particular scenarios constructed; neither provides a comprehensive analysis of the swarm behaviour.
- We aim to apply, assess and develop the use of temporal verification to verify properties of robot swarms.

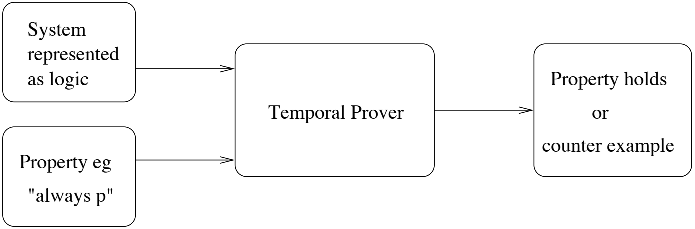
Temporal Verification: Model Checking

- Two main approaches to temporal verification are that of model checking and deductive techniques.
- Model checking is a fully automatic, algorithmic technique for verifying the temporal properties of systems.
- Input to the model checker is a model of the system and a property to be checked on that model.



Temporal Verification: Deduction

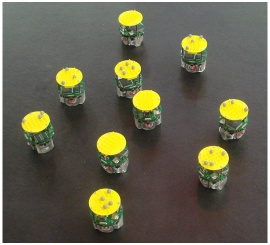
- Deductive techniques involve the representation of both the system and the property as logical formulae and applying mathematical proof to these.
- The logics are some form of temporal logic which has operators that relate to time eg '$\Diamond$' (*sometime in the future*), or '$\Box$' (*always in the future*).


System represented as logic
Property eg "always p"
Temporal Prover
Property holds or counter example

Case Studies

We consider two case studies.

- Firstly we consider the verification of the connectedness property of a particular robot swarm algorithm, the alpha algorithm, which makes use of local wireless connectivity information alone to achieve swarm aggregation.
- Secondly we apply probabilistic model checking to a swarm of foraging robots.



The Alpha Algorithm

- The default behaviour of a robot is forward motion.
- Avoidance behaviour is carried out when robots are near each other.
- While moving each robot periodically sends an “Are you there?” message. It will receive “Yes, I am here” messages only from those robots that are in range, namely its neighbours.
- If the number of a robot’s neighbours should fall below the threshold α then it assumes it is moving *out* of the swarm and will execute a 180° turn (coherence).
- When the number of neighbours rises above α (when the swarm is regained) the robot then executes a random turn.

Link

Our Approach

1. Take the design of a swarm control algorithm for an individual robot.
2. Describe an abstraction that tackles the continuous nature of the domain, the potentially large number of robots, the nature of concurrency and communication.
3. Carry out model checking to assess the temporal behaviour of the model from (2). If model-checking succeeds, then return to (2) refining the abstraction to make it increasingly realistic. If model-checking fails, then analyse the failing trace (by hand). Either there is a problem with the original algorithm, so this must be revised, or the algorithm is correct for this scenario and so the abstraction in (2) must be revisited and expanded to capture this behaviour.

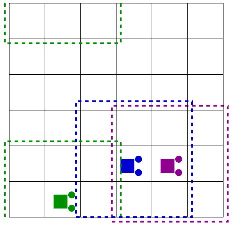
Issues

- The need to abstract away from many details to obtain a discrete and finite representation, eg robot location, direction, wireless range, step size etc.
- The choice of concurrency.
- The need to model uncertain information;
- The size of the state space–impacts on representing the location of the robots, number of robots, directions etc.

Model Checking

- con_i is a derived proposition which is true for robot i if there are at least α robots within its wireless range and false otherwise.
- We model the alpha algorithm and aim to verify $\Box\Diamond con_i$ for each of the i robots.
- Note that with this property isn't exactly what we need as it could hold with the swarm split into more than one connected groups.
- We consider different models of concurrency, different number of robots, grid sizes with different α parameters and wireless range.

Representation



- We use a “wrap round” grid representation with at most one robot in each square and four directions of movement.
- The wireless range is represented as a number of squares from the robot’s position.
- There are two robot modes *forward* and *coherence* and each robot can also be *connected* or not.
- We assume a step size of one grid square and that a robot can detect other robots for avoidance in the adjacent

Concurrency

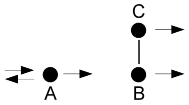
- We considered different concurrency options: fair asynchrony, non-strict turn taking, strict turn taking and then synchrony.
- However the property is false for all grid sizes and number of robots we tried for asynchrony, non-strict turn taking and strict turn taking.
- The failing traces for asynchrony, strict and non-strict turn taking show robot one makes a move resulting in the robots losing connectedness which they never regain.
- Whilst we initially believed synchrony was not the best way to model the swarm due to the slight physical differences between robots etc this appears to be the best abstraction.

Results

We applied model checking to a variety of grid sizes, numbers of robots, wireless range, and alpha parameters but can only deal with a small number of robots and small grid sizes.

Certain (grid independent) types of failing trace can be unfolded into a larger or infinite grid (so no need to check larger grid sizes).

Using model checking we obtain failing traces of the form below.



These results confirm a known problem with the alpha algorithm, when a robot or group of robots is linked to the rest of the swarm by a single link (known as a bridge or cutvertex).

State Explosion Problem

- We would like to consider larger number of robots and larger grid sizes but we are faced by the well known state explosion problem
- Even with the simplifications we use here, the state space explored is huge.
- We can't just keep increasing the number of robots and grid size.
- To combat this we will have to apply and develop some of the work that has been carried out in the model checking field using more
  - sophisticated abstractions,
  - clever representations, and
  - techniques such as symmetry (see Kerstin's recent TAROS paper),

  in order to reduce the state space.

Discussion: Relative Parameter Sizes

- Initially we set the step size, avoidance detection and the wireless detection all to be the same (one unit).
- The intention in the alpha algorithm is that these should be

$$\text{step size} < \text{avoidance detection} < \text{wireless range}$$

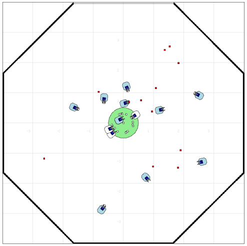
- In increasing the wireless range we have achieved the latter but not the former.
- Similarly we set the *cadence* (periodicity of the detection of other robots) at one and this is intended to be larger.
- These could further be investigated but the state explosion problem will again be a limiting factor.

Verification of Alpha Algorithm: Summary

- The state explosion problem meant we could only consider small grid sizes and a small number of robots.
- The small grid size seems less of a problem as once we have found a grid independent failing trace we can unfold this into a larger or infinite grid.
- Even with different α parameters and wireless range we obtained failing traces that seem to match the documented cutvertex flaw in the alpha algorithm.
- More fine tuning is needed for the relative sizes of the parameters.
- In the above analysis we have ignored any uncertainty, for example the accuracy of the sensors at the limits of the wireless range.

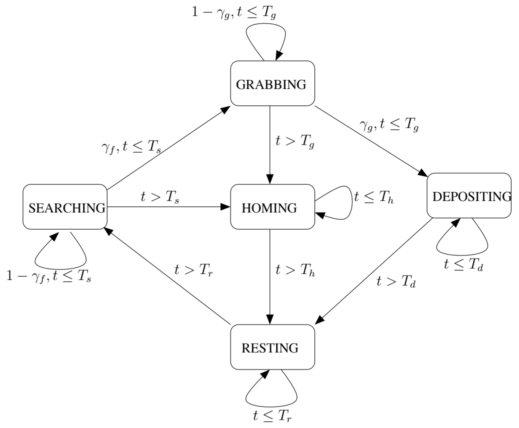
Dealing With Uncertainty: Foraging Robots

- Foraging is a commonly used robot scenario.
- Within a fixed size arena, each foraging robot must search for and bring food items back to the shared 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 energy up and there is no guarantee that robots will actually *find* any food.



A Probabilistic State Transition System

The behaviour of each robot in the system is represented by the probabilistic state transition system below.



Based on W. Liu, A. Winfield, J. Sa, Modelling Swarm Robotic Systems: A Study in Collective Foraging, Proc. Towards Autonomous Robotic Systems (TAROS), 2007 pp. 25–32.

Modelling a Swarm of Foraging Robots

- We use PRISM a probabilistic model checker that supports different types of probabilistic model.
- If we represent each robot as a probabilistic state transition system and take the product of these we can only deal with small numbers of robots (as previously) due to the state explosion problem.
- Instead we use a *counting abstraction* by modelling the system using just one state transition system but with a counter recording how many robots are in each state.
- We investigated the changes to swarm energy relating to varying different parameters and in different scenarios.
- The energy of the swarm at each step can be calculated by adding the energy value of the food collected and decreasing this by the cost of moving and looking for food in each robot mode.

Verification

We used PRISM to verify properties relating to swarm energy and the number of robots carrying out a particular task in a number of different scenarios:

- changing the probability of finding food;
- variable probability of finding food depending on the number of foraging robots;
- variable probability of grabbing food items relating to the number of robots grabbing;
- changing the resting timeout to be a probability related to the number of robots depositing food or returning without food;
- replacing all timeouts with probabilities.

Results

- The simulation mode of PRISM allows us to vary different parameters and explore their affect on the swarm numbers and energy.
- We have varied several parameters and checked a range of properties such as:
  - *if the number of robots is more than n then the average swarm energy exceeds E within t_A time steps.*
  - *after t_A timesteps, if the size of the swarm is greater than k then the number of foraging robots its greater than n*.
- For the same settings of parameters from the original paper we obtain similar results.
- We didn't include any states relating to avoidance but could do this by adding an avoidance state to every state apart from *resting*.

Probabilistic Model Checking: Summary

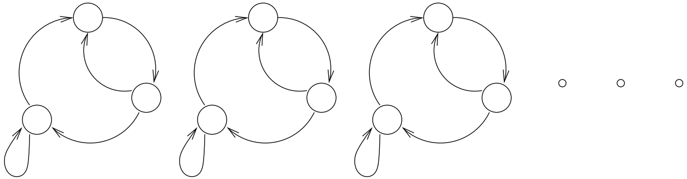
- We explored using a probabilistic model checker to verify properties of an existing probabilistic model of foraging robots.
- We used a counting abstraction to avoid the state explosion problem focusing on the number of robots in each state. This allows us to consider large numbers of robots.
- In particular, we investigated the changes to swarm energy relating to different parameters and scenarios.
- Such an analysis provides not only simulation but also allows verification and thus provides an additional tool for the swarm algorithm designer.
- In a related approach Herd et al. apply a combination of simulation and statistical model checking to swarm verification.

Conclusions

- We have applied standard and probabilistic model checkers to swarm algorithms for coherence and foraging.
- Failing traces in the alpha algorithm confirmed a known issue.
- Probabilistic model checking allowed us to investigate swarm energy with different parameters and scenarios.
- Abstractions are necessary to tackle the continuous nature of the domain and the state explosion problem.
- Swarm verification is not intended to replace real life robot experiments or simulations but to aid the development of principled design techniques for robotic swarms.
- Other approaches include incorporating model checking into the design process (Brambilla et al.) and calculating a lower bound for an emergent property and then model checking using this bound (Kouvaros and Lomuscio).

Current and Future Work

- We would like to apply some of the techniques to deal with the state explosion problem in particular better representations or abstractions.
- We would like to apply model checking to other swarm algorithms (recently we have been looking at Firefly Algorithms).
- Develop further and apply recent advances in deductive methods at UoL to swarm robotics for example in relation to first-order temporal logic.



References

- Dixon, C., Winfield , A.F.T., Fisher, M., and Zeng, C. Towards Temporal Verification of Swarm Robotic Systems, Robotics and Autonomous Systems, 60(11): 1429-1441, Elsevier, 2012
- Konur S., Dixon C., and Fisher M. Analysing Robot Swarm Behaviour via Probabilistic Model Checking, Robotics and Autonomous Systems, 60(2):199-213 Elsevier, 2012
- Winfield, A.F.T., Sa, J., Fernandez-Gago, M-C., Dixon, C., and Fisher, M. On Formal Specification of Emergent Behaviours in Swarm Robotic Systems, International Journal Of Advanced Robotic Systems, 2(4), pages 363-370, 2005