Homework Module: A Controller for Swarm Behaviour in Webots

Aleksander Burkow
Sigve Sebstian Farstad
Emil Grønnbeck

January 27, 2014

Abstract

This report presents a solution for Homework Module #1 of IT3708, spring 2014 at NTNU. The purpose of the homework module is to "understand swarm behaviour by implementing a controller for box pushing task in Webots" [2].

Contents

Introduction

“Swarm robotics is a field that capitalizes on self-organization to generate interesting global patterns from interactions among relatively simple robots, all in the absence of centralized control.” [1]

This report presents three homogeneous robotic swarm that can identify interesting objects and push them to desired areas. The individual agent is inspired by ant behaviour.

Three different versions of the swarm agent is presented: the proposed agent, the improved agent and the advanced agent.

1 Subsumption Architecture

The implemented system is based on an AI concept called subsumption. The subsumption architecture was first described by Rodney Brooks in 1986 [5], and is also called Brooks’ Architecture. The idea behind Brooks’ Architecture is inspired by insect behaviour. Insects, although they have relatively small amounts of computational power, are able to walk, avoid obstacles, and make complex decisions at impressive speeds. The architecture is divided into behavioural layers called: ”the levels of competence”.[6]. Each layer is independent of the others, but the higher levels are capable of overriding the lower. [6] This way, complex behaviour can be composed of many small reactive agent behaviours.

2 Webots

“Webots is a development environment used to model, program and simulate mobile robots.” [3] Webots is used in this assignment to implement and simulate swarm agents.

3 The E-puck

The e-puck is a small (7 cm) open-source differential wheeled puck-shaped educational robot. The E-puck model is available in Webots as a simulatable model, and it is used in this assignment as the hardware on which the swarm agents will run.

Part A
The Proposed System

4 Description

The initial proposed system is based on the subsumption approach suggested in [2]. The ”levels of competence” are divided into six independent behaviours: Wander, Avoid obstacles, Converge, Retrieve, and Reposition. The lower levels have a higher priority and are able to override the higher ones. Figure 1 on the following page shows the subsumption hierarchy.

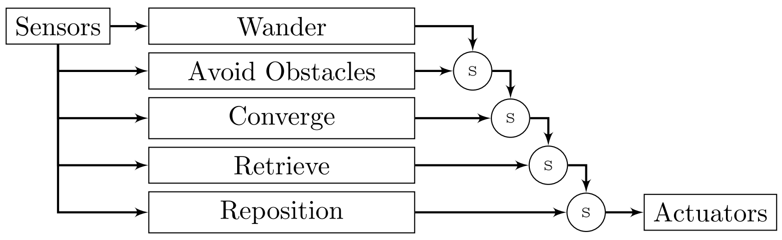


Figure 1: The subsumption architecture of the proposed system.

4.1 Behaviours

Each of the behaviours of the proposed system are independent reactive behaviours that map input sensor data to output actuators. The rest of this section describes each of the behaviours in detail.

4.1.1 Wander

Wander is the default behaviour. It tells the agent to move straight forward as fast as possible, regardless of input.

4.1.2 Avoid obstacles

This behaviour is intended to save the agent from harmful collisions. It is triggered when the agent's proximity sensors rise above a certain threshold, indicating that it's close to or about to hit a wall. The rotational angle of the agent is then set proportional to the difference between the amount of obstacles on either side of the agent. The closer the wall is on the right side, the harder the agent will turn to the left, and vice versa. This technique is called Braitenberg avoidance [4].

4.1.3 Converge

This behaviour is intended to let agents converge on a food location. It is triggered when the sum of the agent's light sensors rise above a certain threshold, indicating that food is nearby. The rotational angle of the agent is set in proportion to the amount of light on either side, making the agent heads towards the food, aligning it with the surface normal of the food for maximum force.

4.1.4 Retrieve

This behaviour is intended to let agents push food to a goal area, and is triggered when the agent detects that food is right in front of it. It will try to align the agent with the surface-normal of the object, such that maximum force will be exerted upon it. It subsumes the obstacle avoidance behaviour, letting the agent push the food.

4.1.5 Reposition

This behaviour is intended to let agents to move out of potentially poor retrieval positions. It is enabled when the agent has found food, and is trying to bring it home.

A problem that commonly occurs is that the agents are trying to push the food from opposing sides, and therefore getting nowhere. The reposition behaviour simply says that after N seconds of retrieval, the agent will try to push somewhere else nearby. This exponential backoff-inspired behaviour allows the agents to reposition themselves reasonably efficiently.

5 Simulation Results

This first version of the swarm agent, dubbed the proposed agent, was run in a series of simulation experiments in Webots. 7 swarm robots were randomly placed in a $1.5m \times 1.5m$ square world together with a randomly placed lit food block. For each experiment, the time taken from world setup was complete until the swarm managed to transport the food to an edge of the world was measured.

This simulation was run 75 times. The results of the simulation can be seen in figure 3a on page 6.

5.1 Observed weaknesses

During observation of the 75 simulation runs, several weaknesses of the agent system were identified; amongst them: low levels of emergent teamwork, and getting stuck.

5.1.1 No incentive for teamwork

In the current architecture, the reposition behaviour blindly sends the robot away after it has been pushing the food for a fixed amount of time. This does not entice teamwork, as robots who by chance are pushing on the same side have equal chance of repositioning themselves as robots who are blindly pushing from some unproductive angle. The result is that the robots don't intentionally cluster together, making food retrieval more luck-of-the-draw based than a coordinated operation.

5.1.2 Getting stuck

In this implementation, the robots had a tendency to get stuck in a few different ways. Fortunately these issues mainly manifested themselves after the pushing of the box into the wall, therefore not interfering with the goal.

The stuck robots would typically get stuck in one of these ways:

- Two robots driving straight into each other
- Getting stuck behind boxes
- Poor sensor calibration sometimes leads to robots not avoiding walls when food is nearby
- Rolling on their sides (rare)

Part B
An Improved System

6 Description

The initial system had some issues with enticing teamwork and preventing robots from having love affairs with one another instead of retrieving food for the starving epuck-population at home.

To resolve these issues, rewards for collaboration when retrieving food was introroduced. Contingency behaviour when stuck was not implemented for the improved system, as agents getting stuck after the food had reached the goal area did obviously not interfere with achieving the goal.

6.1 The improved reposition behaviour

The reposition behaviour has been modified to reward agents working together. This is done by increasing the allotted retrieval-time in proportion to the number of neighbouring robots.

Allowing robots to push for longer periods of time in the vicinity of neighbours tends to converge them together on the same side. Robots trying to go solo will reposition themselves until they realign with the rest of the pack.

The improved behaviour had a noticeable effect on the average running time of our simulations, as shown in the graphs below.

7 Simulation Results

This version of the swarm agent was run in a series of simulation experiments in Webots identical to the simulations in section 5 on the preceding page.

This simulation was run 75 times. The results of the simulation can be seen in figure 3b on page 6.

As hoped, the improved system presents a considerable improvement over the original system presented in part on page 1. Indeed, the median completion time of the improved swarm agent is almost halved from the original agent, bringing it down to a somewhat impressive 21.344 seconds.

Part C
A More Advanced Architecture

8 Description

Exploring the viability of the improved swarm agent from part B in different environments, a two-food scenario is considered. In this scenario there are two food boxes that need to be collected by the swarm.

Adapting the improved swarm agent from part B, a new behaviour is added to the subsumption ladder: an anti-stagnation behaviour called unstick. An overview of the behaviour subsumption scheme for the advanced system can be found in figure 2.

The addition of the unstick behaviour ensures that the agents move back and forth between the two food boxes now and then at random. Whoever is on the same box is working on the same team. This creates an implicit organization into teams that does not rely on sensors or distinguishing actuators such as team colors, making the architecture more resilient to failure.

8.1 Unstick

This behaviour is implemented to introduce jiggle to the system. It is triggered at random, and when triggered, subsumes all other behaviours.

Adding unstick does not affect the single box scenario greatly, as it only introduces random jiggle. The randomness increases the variance of the completion times in a one-box-scenario, but the average completion time remains the same.

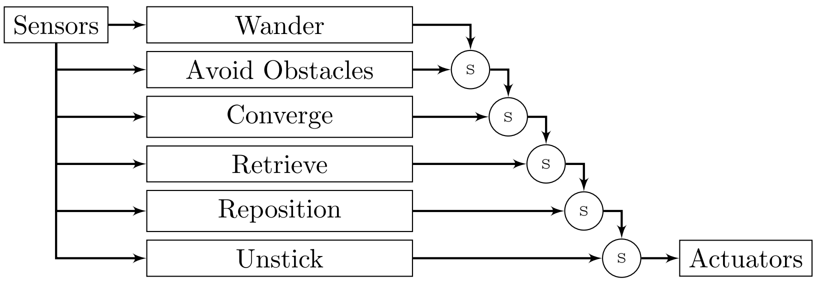


Figure 2: The subsumption architecture of the advanced system.

9 Simulation Results

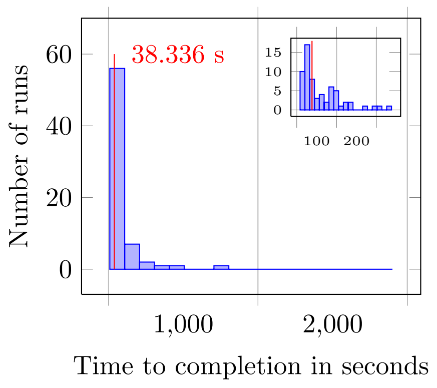
The improved system from part B and the advanced system from part C on the previous page were both run in each their series of two-food simulation experiments in Webots. The simulation setup was nearly identical to the setup in section 5 on page 3, with the exception being that there were two independent but identical randomly placed lit food boxes instead of one.

The simulation was run 75 times with the improved swarm agent presented in part B on the previous page, and 75 times with the advanced system with the unstick behaviour presented in part C on the preceding page. The results from these simulations can be seen in figure 3c on the next page and figure 3d on the following page, respectively.

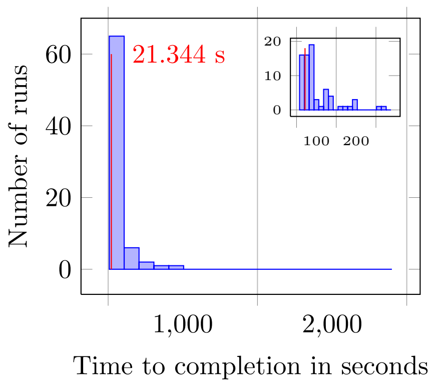
The simulations with the improved agent from part B, without the unstick behaviour, were dominated by runs where the swarm was not able to complete its task. Often the swarm would get stuck in a locked position, and the simulation would have to be aborted prematurely.

In the simulations where the swarm did finish their task, they finished quite quickly - using only one third of the time of the advanced system with the unstick behaviour. This can be explained by survivorship bias.

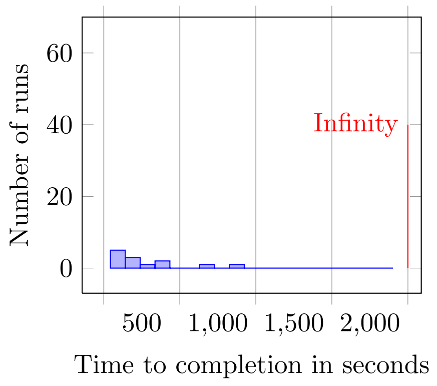
The simulations with the advanced agent from C, with the unstick behaviour, fared much better. The median solution time for the advanced agent in the simulations was 647.872 seconds. Of these seconds, the vast majority were typically spent finding and moving the second food box.



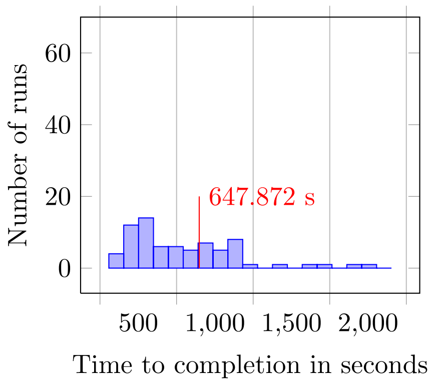
(a) Randomly generated scenario with 1 food box and 7 epuck robots, no incentive to cooperate, without unstick behaviour.



(b) Randomly generated scenario with 1 food box and 7 epuck robots, incentive to cooperate, without unstick behaviour.



(c) Randomly generated scenario with 2 food boxes and 7 epuck robots without unstick behaviour.



(d) Randomly generated scenario with 2 food box and 7 epuck robots with unstick behaviour.

Figure 3: Task completion measurement time distributions for different scenarios. Each scenario was simulated 75 times. The red line indicates the median time. Smaller, embedded graphs are zoom-ins on specific ranges of the data set of the larger set.

10 Conclusion

In this report we demonstrate how purely reactive agents, when amassed, can surmount problems greater than the sum of their parts. They exercise swarm-like intelligence, and through increasingly difficult environments, the internally-stateless warriors prevail. Perhaps there is hope for artificial intelligence after all.

References

[1] - it3708 - subsymbolic methods in ai, spring 2014. http://www.idi.ntnu.no/emner/it3708/.

[2] Homework module: Simulating swarm behaviour in webots. http://www.idi.ntnu.no/emner/it3708/assignments/modules/swarm-project-2014.pdf.

[3] Webots: robot simulator. http://www.cyberbotics.com/overview.

[4] V. Braitenberg. *Vehicles: Experiments in synthetic psychology.* Cambridge, MA: MIT Press, 1984.

[5] Rodney Brooks. A robost layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.

[6] Marissa Warner-Wu. Subsumption architecture. http://mwarnerwu.wordpress.com/research/subsumption-architecture/.