Effective Evolution of Task Specialisation in Multi-Robot Teams

Mostafa Rizk
Faculty of Information Technology, Monash University
Melbourne, Australia
mostafa.rizk@monash.edu

Julian García
Faculty of Information Technology, Monash University
Melbourne, Australia
julian.garcia@monash.edu

Aldeida Aleti
Faculty of Information Technology, Monash University
Melbourne, Australia
aldeida.aleti@monash.edu

Giuseppe Cuccu
eXascale Infolab, Department of Computer Science,
University of Fribourg
Fribourg, Switzerland
giuseppe.cuccu@unifr.ch

ABSTRACT

Foraging is a common task in the robotics domain that involves transporting resources from one location to another. It has analogues to many real world applications such as mining, space exploration and hazardous waste removal. Using a distributed multi-robot team is advantageous as the team is scalable and robust, but designing controllers is difficult. This is especially true when the team divides the task into different sub-tasks and sub-groups need to specialise. Evolutionary algorithms are a promising tool for controller design, however little work has been done evolving controllers for teams performing task specialisation. This paper examines a foraging scenario requiring task specialisation and explores the difference in performance of a heterogeneous and homogeneous team. In the heterogeneous setup, the sub-groups have different controllers and evolution uses individual selection. In the homogeneous setup, the sub-groups have the same controllers but perform different tasks depending on environmental cues, and evolution uses team selection. We provide results suggesting how to most effectively evolve a team that achieves high performance in this variant of the foraging problem that requires specialisation.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent systems**; **Cooperation and coordination**; **Modeling and simulation**;

KEYWORDS

Evolutionary algorithms; Cooperation; Division of labour

ACM Reference Format:
Mostafa Rizk, Julian García, Aldeida Aleti, and Giuseppe Cuccu. 2019. Effective Evolution of Task Specialisation in Multi-Robot Teams. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

. As we automate more industries it will become more commonplace to have teams of robots cooperating on task. Programming these robots can be tedious and difficult. Evolution provides us with a way of generating neural network controllers for multiple robots cooperating to achieve a shared goal, but it remains challenging to produce robot teams that exhibit specialisation i.e. different members of the team perform different sub-tasks. In other words, it is still not fully understood what is the best way to perform task partitioning and task allocation when evolving a multi-robot team.

. In previous works, researchers have evolved teams that are either homogeneous (all robots have the same controller) or heterogeneous (different sub-groups have different controllers). In addition to this distinction, in the evolutionary process, selection is either performed at the individual level or team level. At the individual level, each robot in the team is assessed independently whereas at the team level, they are assessed together. This is illustrated by Waibel et al. in [Waibel et al. 2009]. However, in that work, Waibel et al looked at tasks that did not benefit from specialisation. In this study we compare these 4 setups but consider a foraging task where the team has higher performance if they perform it in a specialised way.

2 EXPERIMENTAL SETUP

2.1 Simulation

. We use an experimental setup inspired by the work of Ferrante et al. [Ferrante 2015]. Ferrante et al proposed a task where a team of robots forage for resources placed at the top of a slope. The setup is intended as an analogue for leafcutter ants retrieving leaves from the top of a tree, where there is a time cost associated with climbing to where the leaves are. Leafcutter ants however, have evolved to leverage gravity, with the ants partitioning the task and some of them climbing up, cutting leaves and dropping them in a cache at the base of the tree and some of them, which can be thought of as generalists, collecting leaves from the cache and returning them to the nest and some of them occasionally cutting the leaves and transporting them back themselves. Specialist teams of ants turn out to be more efficient because they do not lose time repeatedly climbing up and down the tree. Similarly, robot teams that evolve to use specialisation retrieve more resources in the same amount of time as their generalist counterparts. In this task, there is evolutionary pressure for robot teams to develop a specialist strategy and Ferrante et al show that it is possible for such specialisation to emerge through artificial evolution much as it did in natural

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. …$ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
…$15.00

evolution. We use a simplified version of Ferrante et al's setup that uses simple agents rather than a physicall accurate robotics simulation as the focus of this study is the evolutionary conditions required for specialisation to emerge, not necessarily the evolution of real world-ready robot controllers. The simplified simulations allow for more thorough study of evolutionary conditions without the additional complications of realistic physics, sensor noise and high computational cost.

. The arena is 8 tiles long and 4 tiles wide. The bottom row of tiles is the nest, where robots initially spawn and where they must return resources. The next 2 rows are the cache, where resources that have been dropped come to rest after sliding down the slope. The next 4 rows are the slope, which resources slide down when dropped. The final row is the source, where resources spawn at the beginning of the simulation and where new resources spawn when older ones have been removed. Initially 3 resources spawn in the source and 2 robots spawn in the nest. The robots' goal is to transport as many resources as possible to the nest. In the implementation by Ferrante et al, the fitness of the team of robots is how many resources they retrieved. In that implementation, robots take longer to travel up the slope than to travel down and resources slide down faster than robots. This is difficult to replicate without a realistic physics engine and in a discrete environment. To achieve a similar effect, we introduce a cost for expending energy, much like a battery. Traveling up the slope has a higher cost than traveling down. This is an accurate reflection of real robots which would expend more power going up a slope and also produces evolutionary pressure for the robots to specialise.

. Robots can move forward, backward, left and right and can pickup or drop a resource. A resource must be in the same tile as a robot to be picked up. Robots can sense all 8 adjacent tiles and the tile they occupy. They can detect whether the tile is blank, contains a resource, contains another agent or a "wall" i.e. a tile beyond the confines of the arena. They can also detect which of the 4 areas they are on and whether or not they're carrying a resource. Robots are controlled by a recurrent neural network with 41 inputs (9 tiles x 4 possible tile contents + 4 possible locations + 1 boolean bit indicating resource possession), 6 outputs and no hidden layers.

. Each time step, the agents make their observations and use the neural network to choose an action. If an agent moves, it moves one tile per time step. If an agent picks up a resource, that resource moves with it. If it drops a resource, the resource is placed in the tile the robot is currently in. If the resource is dropped on the slope, it slides down the slope at a rate 10 times smaller than the slope angle. That is, if the slope angle is 20, the resource moves 2 tiles per time step until it reaches the cache, at which point it comes to rest. Resources cannot slide beyond the cache, they stop just before reaching the nest.

2.2 Evolutionary Configurations

. We compare homogeneous and heterogeneous performance for 5 different slope angles: 0°, 10°, 20°, 30° and 40°.. For each we perform 30 evolutionary runs of CMA-ES using random seeds 1-30, making a total of 300 runs (30 random seeds x 2 team compositions x 5 slope angles). Each run consists of several iterations of CMA-ES, where the genome being evaluated is a list of neural network weights for the agent controller (two controllers if it is a heterogeneous team). In an individual iteration, the genome's fitness is evaluated through 10 trials of the simulation, where the simulation is 1000 time steps long. Each run of CMA-ES has sigma value 0.05 and is seeded with an initial genome produced by a random weight guessing (RWG) algorithm. RWG randomly guesses genomes for 5000 tries, terminating prematurely if one of the genomes has a fitness greater than or equal to 3 (i.e. 3 more resources are retrieved by the team on average over all 10 trials). If no such genome is found in the 5000 tries, the best one found so far is used.

. We use CMA-ES as the evolutionary algorithm to find a solution. Before CMA-ES is run, RWG is run to find a genome with fitness slightly above the minimum and it is used as a seed for CMA-ES. This initial random search helps avoid CMA-ES getting stuck in a flat fitness landscape.

. RWG is run for 500 generations or until a genome with fitness > 100 is found, and the fittest genome is returned as a seed for CMA-ES. A genome is a vector of weights to be loaded into one or more neural networks. The length of the genome varies depending on which of the four below configurations is being used.

. In CMA-ES, a population of 40 individuals is generated and the algorithm is run for 5000 generations. Each individual is a genome. The fitness of an individual is calculated by doing 5 runs of the simulated environment, each lasting for 500 time steps. A neural network is created and it loads the W-weight vector. At each time step, the observations of each agent are independently fed to the neural network which in turn provides the action to be taken. The actions are input to the environment and used to generate the state at the next time step. At each time step,

2.2.1 Homogeneous Team with Team Selection.

. A genome is the weights for one individual. There are 40 teams. 40 genomes are created. All robots on a team get the same genome. All members of the team share the reward and cost. A fitness value is returned for the whole team. 40 fitness values are returned for the whole generation. The genome used by the best performing team is selected.

2.2.2 Heterogeneous Team with Team Selection.

. A genome is the weights for two individuals. There are 40 teams. 40 genomes are created. Half of the team get the first half of the genome (the weights for one neural network) and half of the team get the second half of the genome. All members of the team share the reward and cost. A fitness value is returned for the whole team. 40 fitness values are returned for the whole generation. The genome used by the best performing team is selected.

2.2.3 Heterogeneous Team with Individual Selection.

. A genome is the weights for one individual. There are 40 teams. 80 genomes are created since a team needs two genomes. For every two genomes, half the robots on the team get the first genome and half the robots on the team get the second. Team members are evaluated separately. They share the reward but bear costs

separately. A fitness value is returned for each team meber. 80 fitness values are returned for the whole generation. The genome used by the best performing individual on any team is selected.

2.2.4 *Homogeneous Team with Individual Selection.*

. A genome is the weights for one individual. There are 40 teams. 80 genomes are created where every second genome is a duplicate of the previous genome i.e. genome 2 is a copy of genome 1, genome 4 is a copy of genome 3 etc. For every 2 (identical) genomes, half the robots on a team get the first and the other half get the second. All members of the team share the reward but bear the costs separately. A fitness value is returned for each individual. 80 fitness values are returned for the whole generation. The genome used by the best performing individual is selected.

REFERENCES