Artificial Life and Robotics (2020) 25:569–577
https://doi.org/10.1007/s10015-020-00650-2

ORIGINAL ARTICLE

Evolving behaviour trees for supervisory control of robot swarms

Elliott Hogg[1] · Sabine Hauert[1] · David Harvey[1,2] · Arthur Richards[1]

Received: 20 May 2020 / Accepted: 13 September 2020 / Published online: 18 October 2020

Abstract
Supervisory control of swarms is essential to their deployment in real-world scenarios to both monitor their operation and provide guidance. We explore mechanisms by which humans can provide supervisory control to swarms to improve their performance. Rather than have humans guess the correct form of supervisory control, we use artificial evolution to learn effective human-readable strategies. Behaviour trees are applied to represent human-readable decision strategies which are produced through evolution. These strategies can be thoroughly tested and can provide knowledge to be used in the future in a variety of scenarios. A simulated set of scenarios are investigated where a swarm of robots have to explore varying environments and reach sets of objectives. Effective supervisory control strategies are evolved to explore each environment using different local swarm behaviours. The evolved behaviour trees are examined in detail alongside swarm simulations to enable clear understanding of the supervisory strategies. We conclude by identifying the strengths in accelerated testing and the benefits of this approach for scenario exploration and training of human operators.

Keywords Human swarm interaction · Swarm · Artificial evolution · Supervisory control · Behaviour trees

1 Introduction

Growing interest in the use of swarm systems has led to new questions regarding effective design for real-world environments [19]. Although their use has great potential in areas ranging from search and rescue to automated agriculture, there are still few examples of real-world deployment. Human supervision has been proposed to improve the performance of swarming by maintaining the scalability and robustness of swarms whilst taking advantage of human intelligence [4].

Human swarm interaction (HSI) aims to adapt swarm models into hybrid systems which operate with the aid of a human operator. The operator can compensate for the limitations of swarming behaviours and increase performance by interacting with the swarm. This is achieved through the human's higher-level understanding and reasoning of a task that individually, swarm agents cannot perceive. This control scheme allows the human to correct for poor performance and research has highlighted significant improvements over purely autonomous swarms [5].

In HSI, research has investigated ways in which an operator can infer high-level knowledge to the swarm to improve performance. Initial answers have defined methods to directly control how the swarm behaves. If the operator has information that the swarm cannot perceive, then the operator can take action to account for the swarm's lack of knowledge. These types of control have been categorized into four different types as discussed in Kolling's survey on HSI [12].

- Algorithmic: changing emergent behaviour of the swarm [10].
- Parametric: adjusting characteristics of behaviour [9].
- Environmental: highlight points of interest [5].
- Agent control: to influence its neighbors [20].

Each control method has shown the ability to positively affect the swarms performance in varying scenarios, however, it is difficult to identify the best choice of control methods for a given scenario. We need to consider how the type of application and conditions should influence the way we

This work was presented in part at the 3rd International Symposium on Swarm Behavior and Bio-Inspired Robotics (Okinawa, Japan, November 20–22, 2019).

✉ Elliott Hogg
elliott.hogg@bristol.ac.uk

[1] Bristol Robotics Laboratory, University of Bristol, Bristol, UK

[2] Thales UK, Reading, UK

choose to control the swarm. Other factors that need consideration include cognitive limitations of the operator [11], the optimal timing for human interaction [17], the operators knowledge of swarm dynamics [8], and how the level of interaction affects the robustness of the swarm [21]. Due to the high dimensionality of these scenarios it is difficult to understand how HSI systems should be designed.

There is currently no way to explore supervisory control of swarms systematically. HSI studies are currently investigated using human trials. This is beneficial for understanding how humans behave when learning to control swarms, however, here we focus on exploring the broader context of swarm control and how to systematically design HSI systems. Rather than having the human guess the rules, we propose artificial evolution of decision-making to act as a surrogate for the human. In this investigation, we generate a swarm supervisor which can view and control the swarm at accelerated simulation speed. Using this approach we can exploit the ability to test rapidly in different simulated conditions to gain a better understanding of HSI strategies. Solutions can be used to infer training of human operators and also inform our knowledge of swarm control across a large set of scenarios.

Behaviour trees (BT) models have been explored to represent evolved supervisory control strategies. BTs have been chosen for their human readability and modularity. Trees can be modified and reordered without any need to redesign individual nodes, this is useful for artificial evolution where structures can be modified through crossover and mutation [7, 14]. A BT is a hierarchical model which consists of actions, conditions, and operators connected by directed edges[18]. BTs can represent many decision-making systems, such as finite state automata and subsumption architectures. An example tree is shown in Fig. 1. These trees are modular and can be constructed from a set of action and condition functions which are independent of order. This makes them well suited to artificial evolution. Refer to Colledanchise and Ogren's book for greater coverage of behaviour tree concepts [3].

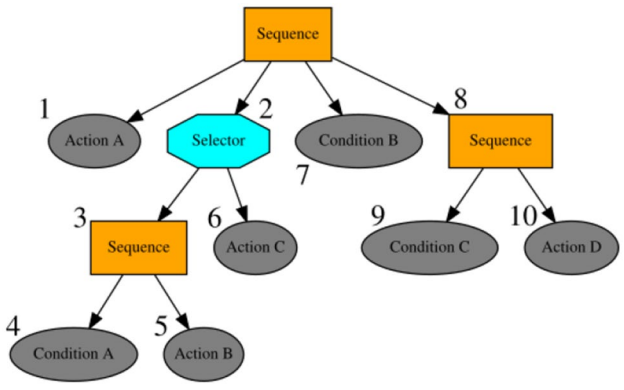

Fig. 1 An example behaviour tree. Numbering shows the order in which the tree is ticked

The following section will explore how artificial evolution has been applied to produce control strategies with BTs to solve a set of coverage based tasks. This problem has previously been explored using other swarming approaches [15, 16]. We then analyse and discuss our findings in section 3.

2 Methodology

The following section will detail how artificial evolution has been applied to develop swarm control strategies for a search and rescue (SAR) scenario. In this work, we use BTs to represent a particular strategy to control the swarm that is evolved to increase task performance.

2.1 Simulation architecture

A custom simulator built-in Python is used to explore 2-D environments with a specified swarm size, and areas of interest to explore. The agents are modelled as particles which can move in any direction with a degree of random motion, and thus, the swarm's behaviour is not deterministic. Each agent follows a trajectory which is dependant on the emergent behaviour of the swarm. Potential field forces act as additional vectors to avoid obstacles and the bounds of the environment. The agents travel at a speed of 0.5 m/s. This framework has been deployed on a super-computer cluster which has allowed us to evaluate many scenarios in parallel. This system is beneficial as we can rapidly assess new conditions simultaneously.

2.2 Scenario

A swarm of ten agents is tasked to search an environment to find a set of objectives as quickly as possible. We investigate performance over a range of environments and objective positions as shown in Fig. 2. For each map, we test for each different set of objective positions, resulting in a total set of 15 unique scenarios. This task is challenging as the total area is sufficiently large in relation to the swarm size and will require efficient coverage to find the objectives quickly. The swarm supervisor has no knowledge of where the objectives are located or the shape of the environment. This will require the swarm supervisor to identify the prior knowledge of the objective positions and nature of the environment. For each evaluation the swarm is initialized in a random starting position within a 3 m × 3 m area at the origin.

The task of the supervisor is to utilize a set of emergent behaviours based on the observation of the swarm state to efficiently search for the objectives. We will now discuss the design of the supervisor in greater detail.

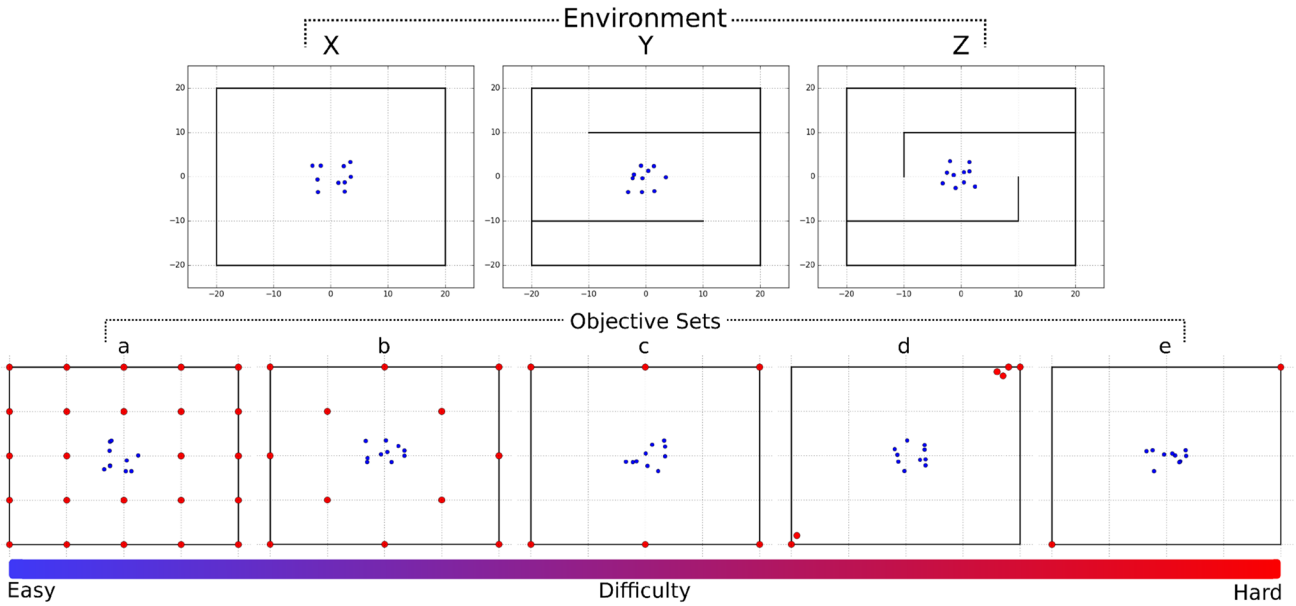

Fig. 2 Environments and objectives under investigation. The starting position for the swarm is shown in blue. Objectives are shown in red. The area of each environment is 40 m × 40 m

2.3 Swarm supervisor

To produce control strategies, a set of actions and decisions need to be defined that will allow the supervisor to interact with the swarm.

2.3.1 Actions

The actions allow the swarm supervisor to change the swarms behaviour. Only forms of algorithmic control are considered which are commonly used in HSI systems as it is easy for untrained operators to understand and enables effective control of the swarm without reducing swarm autonomy [1].

The first behaviour available is *dispersion*. When initially clustered together, agents are repelled by an exponential force from one another to spread into open spaces. Each agent is repelled from every other agent by an exponential force represented as a vector. The sum of these vectors determines the direction of travel. When significantly spread out agents will travel with a random walk [6, 16]. Dispersion was chosen as a natural means for spreading out and filling an area which is suitable for a SAR based task.

The remaining set of actions are defined as *directed fields* which cause the swarm to disperse in a specific direction. The swarm will travel in the specified direction whilst avoiding obstacles and repelling from nearby agents. We enable eight forms of this behaviour such that the swarm can be directed north, south, east, and west. As well as, north west, north east, south west, and south east. These behaviours can be used to direct the swarm to particular regions. The operator may have better knowledge of the area to cover and decide to direct the swarm using this behaviour. Good use of these behaviours relies on the correct choice of the direction of travel.

2.3.2 Conditions

In order for the swarm supervisor to decide on which action to take, knowledge of the swarm state is required. The swarm supervisor can observe the center of mass and spread of the swarm to understand it's approximate location and a sense of formation through its spread. Using the center of mass the supervisor can direct the swarm to specific regions where objectives are located. Understanding spread will enable solutions which promote the swarm to spread out to gain greater coverage.

This high level representation means we don't need knowledge of the whole swarm to enable control. This may be more suitable for human control by reducing cognitive load and scenarios where we don't have complete swarm visibility [22]. The *center of mass* of the swarm is calculated as the average over all agent positions in the x direction μ_x and y direction μ_y, where n is the total number of agents. Each agent ordinate is defined as x_n and y_n. *Spread*, σ, is defined as the average distance from agent to agent as shown.

$$\mu_x = \frac{1}{n}\sum_{1}^{n} x_n, \quad \mu_y = \frac{1}{n}\sum_{1}^{n} y_n \tag{1}$$

$$\sigma = \frac{1}{n * (n-1)} \sum_{k=1}^{k=n} \sum_{i=1:i\neq k}^{i=n} ((x_i - x_k)^2 - (y_i - y_k)^2)^{\frac{1}{2}} \tag{2}$$

Decisions can be constructed using these metrics with respect to defined thresholds shown in Table 1. We enable simple decisions to be made by comparing the real-time metric value to a set threshold. The thresholds that can be selected for the center of mass are bounded within the size of the environment, and similarly, the spread is limited up to the highest level that they can disperse.

We can represent these actions and conditions in node form that can be executed within a BT. At each time step within the simulation, the BT is ticked such that new decisions and actions can be made on the updated swarm state. Evolution will aim to find the optimal combination of decisions and actions for each of the scenarios in our investigation.

2.4 Evolving the swarm supervisor

This section will further detail the design of the swarm supervisor and the evolutionary algorithm used to explore possible solutions within the problem space.

2.4.1 Genetic programming

We use Genetic programming (GP) to evolve BTs [7, 14]. The methodology of GP is to evolve the structure of computer programs represented in the form of hierarchies [13]. We apply common practices in GP to evolve BTs in this investigation. As BTs are modular, we can modify the tree structure using GP and still produce a valid tree. This makes them applicable to evolution which is reliant on random manipulation of genomes. Here we apply the use of single-point crossover, node mutations, and random sub-tree growth.

We define certain constraints on the types of trees that can be generated as well as bounds for mutation. Trees generated for the initial population have a maximum depth of two levels down from the root of the tree. We also limit the number of possible children that an operator may have between 2 and 5. Given the complexity of this search scenario and the number of node choices, we expect that the sizes of trees evolved should match these constraints. Despite this, trees can grow to further depths through crossover and random sub-tree growth. Table 1 represents the available nodes to construct trees and the limits that the conditional statements can take. The evolution is bounded to these limits when generating random trees and performing mutations.

2.4.2 Evolutionary algorithm

We evaluate the fitness of individuals based on the number of objectives that are found and the time taken to find them. An objective is classified as found when an agent falls into a 5 m radius of its position. The maximum reward when detected is 1 point which decays over the duration of the search. The reward decay function is defined by Eq. 3 where, t is the current time-step and dist, is the distance between the objective position and the origin.

$$\text{Objective reward} = 0.95^{\text{t/dist}} \tag{3}$$

$$\text{Fitness} = \frac{1}{\text{total objectives}} \sum \text{objective rewards} \tag{4}$$

This decay function reduces the reward associated with each individual objective over time, therefore, promoting fast search strategies. The rate of decay is dependent on the distance of the objective from the starting position of the swarm. This means that objectives that are easy to find and require less exploration have a faster decay. Similarly, objectives that are further from the origin decay slower. The individual fitness is defined as the summation of all objective rewards collected over each search normalized against the total number of objectives as shown by Eq. 4. This fitness is averaged over 5 attempts for each individual. We also add a

Table 1 The limits defined by the GP algorithm for the types of nodes that can be selected to produce BTs

Node type	Selection choices
Operator	Selector/Sequence (Between 2–5 children)
Action node	Dispersion/North/South/East/West/North East/North West/South East/South West
Condition node	μ_x >-18, μ_x >-16, μ_x >-14, ... increment by 2 ..., μ_x >14, μ_x >16, μ_x >18
	μ_x <-18, μ_x <-16, μ_x <-14, ... increment by 2 ..., μ_x <14, μ_x <16, μ_x <18
	μ_y >-18, μ_y >-16, μ_y >-14, ... increment by 2 ..., μ_x >14, μ_y >16, μ_y >18
	μ_y <-18, μ_y <-16, μ_y <-14, ... increment by 2 ..., μ_x <14, μ_y <16, μ_y <18
	σ >1, σ >2, σ >3, ... increment by 1 ..., σ >14, σ >15, σ >16
	σ <1, σ <2, σ <3, ... increment by 1 ..., σ <14, σ <15, σ <16

condition that limits tree size to contain at most 35 nodes. In this case, evolution aims to maximize fitness.

Tournament selection is used for groups of three individuals followed by single-point crossover, single-point mutation, and sub-tree growth with probabilities shown in Table 2. Elitism is used to save the best 10 individuals from each generation of 100 individuals. For each individual, we set a time limit of 800 seconds. This time limit is significantly long to allow the swarm to fully explore the area.

3 Findings and analysis

We have evolved control strategies in 15 different scenarios based on all pairwise combinations of environments and objective sets (Fig. 2). We will discuss the fittest individuals evolved for each scenario (Fig. 3) and their approach to each problem (Fig. 4). We represent the behaviours through trails formed by the swarm agents as they search the environment. The colour of the points indicate the position in time, initially plotted as blue and shifting over time towards red when reaching the time limit. Objectives that have been detected are indicated by green crosses and red circles when undetected.

Table 2 Evolutionary parameters

Parameter	Value
Population size	100
Test limit	800 s
Elitism size	10
Tournament size	3
Single point mutation probability	0.1
Sub-tree growth probability	0.05

In all cases, the evolved solutions use combinations of different swarm behaviours whilst observing the swarms state to maximize performance. In no case were single behaviours used. When compared to a baseline of the swarm performing the dispersion behaviour under no supervision, the addition of evolved supervision provided a significant increase in performance (Fig. 5). For the majority of solutions, the directed field behaviours were favoured over the use of pure dispersion which was shown to be less effective. Solutions used combinations of center of mass and spread to control the swarm whilst in some, using only one of the metrics. For each scenario, we see that solutions are correctly specializing to each task and achieving high performance. Differences in strategies are largely dependant on the type of environment used, with additional influence based on the objective sets.

3.1 Scenario specific solutions

For open environment *X*, all solutions scored highly. We see that solutions *b* and *c* formed similar strategies to sweep across the entirety of the environment to find all objectives. Both solutions used spread to control the swarm and their behaviours are the same despite the direction of the sweep and tree structure. For objective sets *d* and *e*, we see the solutions direct the swarm specifically to the corners where the objectives are located rather than performing a full sweep. Because of this, they score highly by directing the swarm along the shortest path to the objectives and using center of mass to change directions. For objective set *a*, not all of the objectives are found and a similar behaviour to *d* and *e* is formed rather than a complete sweep to find all objectives.

For map *Y* we see more complicated behaviours with up to 4 phases of movement. In scenarios *a* through to *c*, we see very good performance as full coverage of the environment is achieved. Each of these behaviours have 3–4 phases

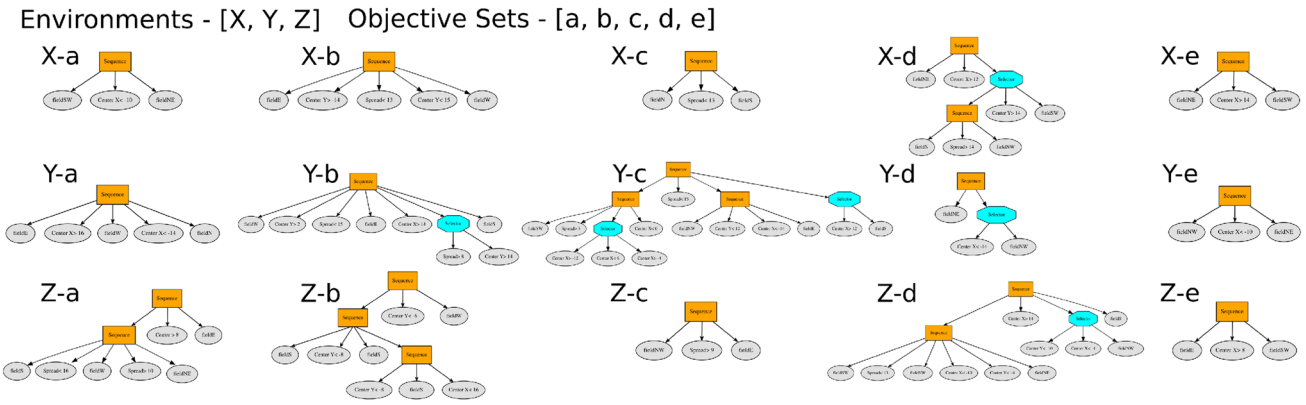

Fig. 3 The fittest individuals evolved in each scenario. We show condensed versions of the trees by removing redundant parts which are not activated. This is a common issue in GP and will be addressed in future work

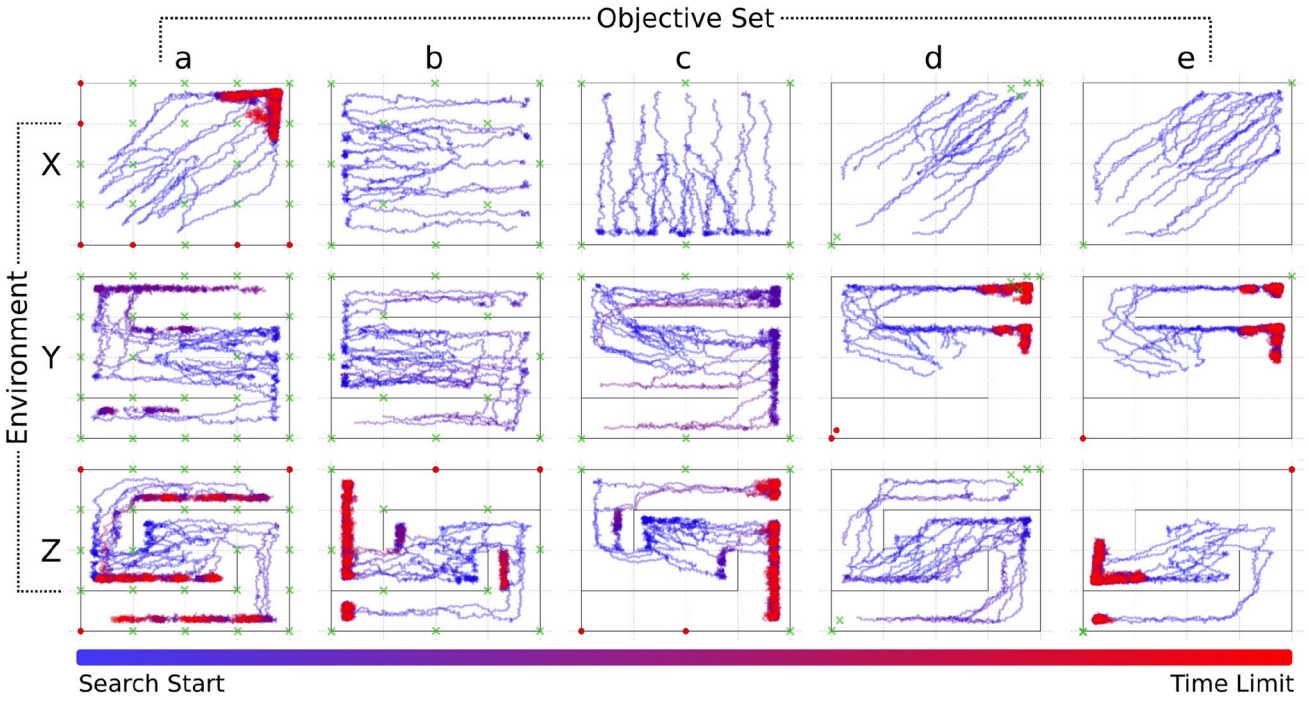

Fig. 4 Trails of the fittest solutions under their trained scenarios. The colour of the points represents the position in time. Initially blue, and shifting to red when approaching the time limit. Detected objectives are highlighted with green crosses, red circles indicate undetected objectives

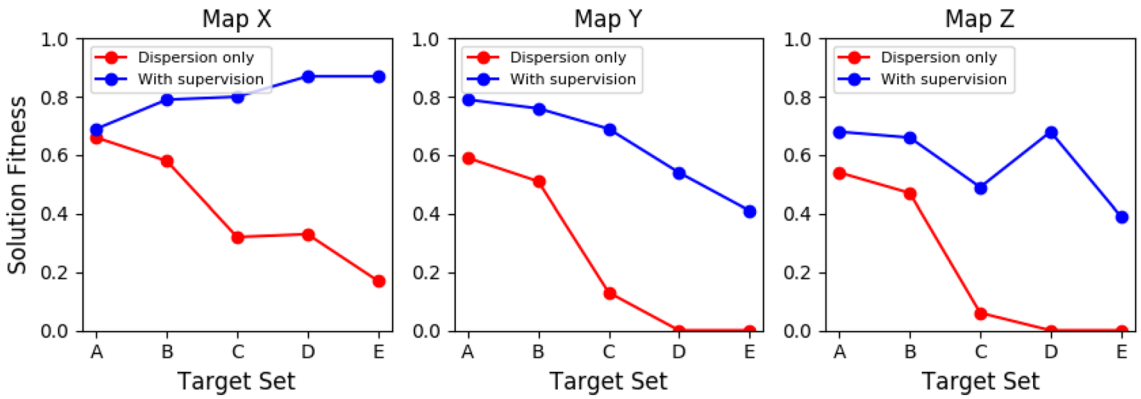

Fig. 5 Performance of the evolved solutions across each map and target set. A baseline of dispersion under no supervision is also presented. Each point represents the mean fitness produced over 300 trial runs

of movement and use a combination of center of mass and spread to direct the swarm. Solution *b* sends the swarm west, east, south, and finally west to maneuver around the walls. In cases *d* and *e*, the swarm is directed to the corners where the objectives are placed but don't attempt to reach both corners, subsequently achieving lower scores.

In map *Z*, we see that the solutions score lower overall than the previous map due to further increased difficulty. In scenarios *a* to *c*, good coverage is formed but some objectives are not found. Solution *e* again only attempts to reach one objective as appose to both. However, solution *d* generates very good coverage and is able to reach both ends of the environment. We also see interesting behaviour where the swarm is split between the walls to reach each end of the map faster. This approach is seen across many solutions in map *Y* and *Z* and highlights that these are useful, non-obvious solutions.

3.2 Human understandable strategies

In addition to observing the control of the swarm, we can track the states of each BT and learn how the behaviour is formed. We have the ability to animate these BTs in real-time to quickly identify when nodes are triggered. Figure 6

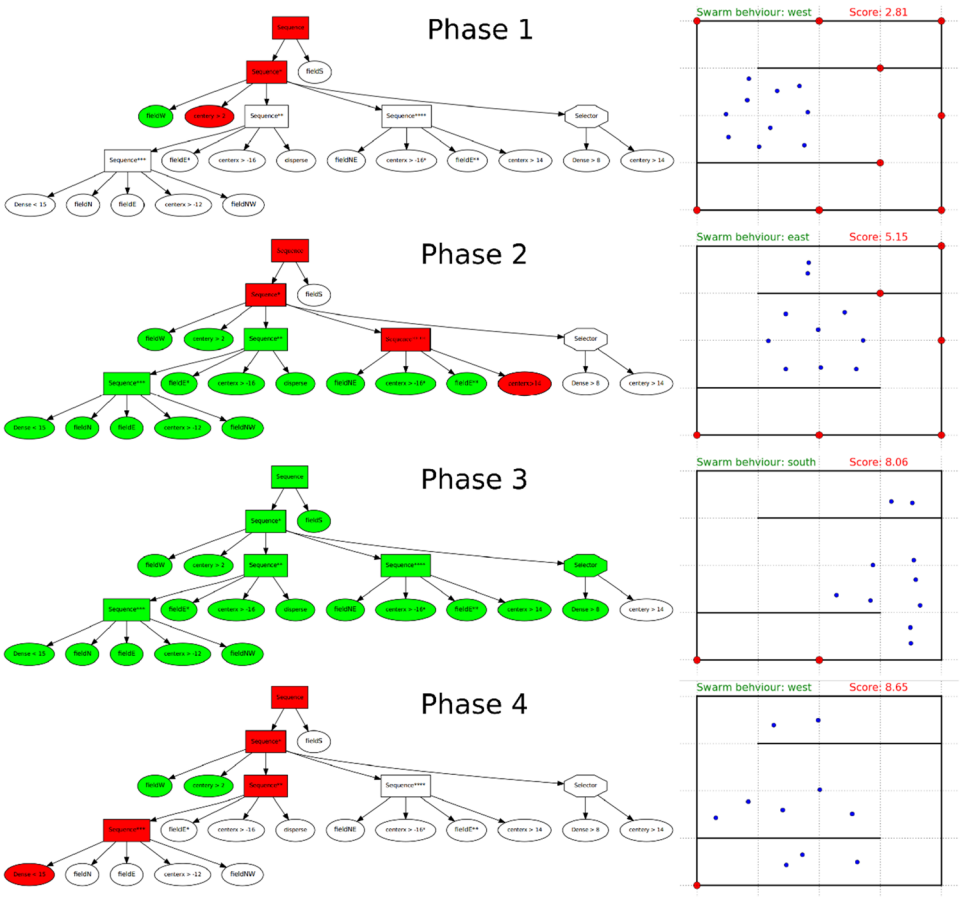

Fig. 6 The 4 stages of decision making in solution *Y-b*. This is achieved in real-time with tools we have created to animate the states of the BT

shows the key states of the BT that controls the swarm in scenario *Y-b*. The colours of the nodes indicate the state of the node, green for success, red for failure, and white if they have not been triggered.

With this visualization, we can break the strategy down into easy to understand key stages of commands. Initially, the swarm is directed west until the center of mass is greater than 2 in the y direction. This enables the swarm to access the upper corridor. Using the y center of mass, the tree observes when the swarm reaches the end wall and has caused some agents to head north. Once triggered, the swarm heads east in phase 2, leaving some agents in the top corridor and some in the central corridor. This is an interesting approach to reach the top corridor whilst simultaneously getting reading to explore the bottom corridor. Once the center of mass is greater than 14 in the x direction and the swarm is at the right-hand side wall, the tree directs the swarm south in phase 3 to enter the bottom corridor. Finally, when the spread is no longer greater than 15, the swarm heads west in phase 4 to travel down the last corridor. This visualization makes it very easy to understand the key decisions that are made to change the swarms movement.

3.3 Scenario coverage

Based on the behaviours of the evolved solutions we see that strategies specialize to their trained scenarios. We also assessed each solution in all other scenarios to see whether solutions could generalize to other scenarios. We summarize the performance of all solutions in Fig. 7. The trained solutions are shown by rows and the columns denote which scenario they were tested against. Each tile shows the average performance of each evolved solution in a given scenario. Each tile represents the fitness of the individual averaged over 300 trial runs.

We observe that for map X it is easy to achieve high scores where as for maps Y and Z the scores decline due to increasing difficulty whilst still performing well. Similarly, the performance declines when moving from objective set a towards e. If we consider the performance of solutions in different maps we see the expected trend that solutions trained in map X do not perform well when assessed in maps Y and Z. However, solutions trained in maps Y and Z perform far better when applied to other environments. Because of this, we see that overall higher fitness falls under the leading diagonal of the heat map.

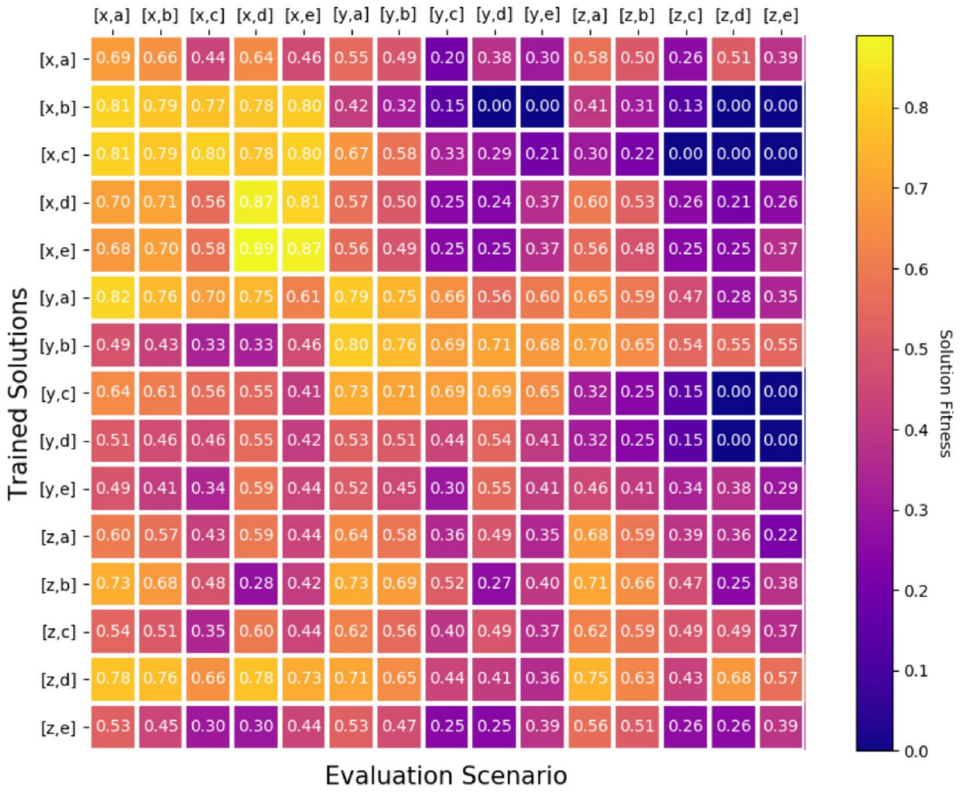

Fig. 7 We evaluate each evolved solution (rows) against all scenarios investigated (columns). High performance is associated with higher fitness

We also identify a number of solutions that generalize well across many scenarios. In particular, *Y-a*, *Y-b*, and *Z-d*. We see that a solution trained on scenario *Y-a* performs well on all map *X* and *Y* scenarios, with a slight decline in map *Z*. Alternatively, *Y-b* struggles in map *X* but performs very well in map *Y* and map *Z*. *Y-b* does not perform well in map *X* as the first decision making step is fitted to the fact that there are walls in environment *Y*. The tree directs the swarm west and waits for the condition, $\mu_y > 2$, to trigger before moving to the next action. Without a wall, μ_y will approximate to zero when heading west and reaching the wall. Hence no further action will be triggered resulting in low performance when executed in map *X*. Alternatively, *Y-a* similarly sweeps east and west but relies only on observing when the swarm reaches each end wall using μ_x to decide when to change direction. This means that the strategy relies on the directed field behaviours and the autonomous nature of the swarm to navigate around any obstacles rather than creating a specialized strategy that learns the nature of the environment. This approach is a good balance of supervisory control whilst taking advantage of the emergent properties of the swarm.

From this finding, we can also identify that spread is more affected by different environments in comparison to center of mass. We see similar sweeping behaviours evolved in map *X* as those created by *Y-a* but using spread as the main decision metric, however, these kinds of strategies perform poorly in the walled environments. This is because the swarms spread is dependent on the shape of the environment whilst certain conditions using center of mass are less effected such as those in *Y-a*. Therefore to produce solutions that generalize, center of mass is a more suitable metric in comparison to spread.

4 Conclusion and future work

Our results showed that we could produce strategies which generated systematic environment coverage and specialized to their scenarios. This was achieved through observation of the swarm state and identifying the prior knowledge of the objective locations. In this case, we were able to explore a wide set of conditions that could be used to control the swarm and explore how this affects the types of strategies that are evolved. Because of this, we were able to identify a number of solutions that generalize as an outcome of these scenarios. The ability to continuously develop swarm control strategies through artificial evolution makes this exploration of many different scenarios and constraints rapid and systematic.

We expect that the evolution of supervisory strategies will be a useful tool to help train human operators by inferring potential solutions and predicting the best approach for certain scenarios. It has been highlighted that an operators understanding of swarm dynamics and time spent controlling swarms can have significant affects on the swarms

performance [8]. By presenting solutions that we have artificially evolved, operators can use these solutions as a starting strategy and if needed, make adjustments if improvements can be made. We also consider that the summarised swarm view in this investigation could be useful to further reduce cognitive load for operators. This idea has been explored in Becker's work on HSI using crowd-sourcing techniques [2].

Future work will aim to explore more complex indoor environments which could accurately represent real-world problems. Further to this, we will implement other forms of swarm control present in HSI to observe how an evolved supervisor can use varying forms of interaction to produce novel solutions.

Acknowledgements This work was funded in partnership between Thales Group and the University of Bristol, and UK Engineering and Physical Sciences Research Council Grant Award EP/R004757/1 entitled "Thales-Bristol Partnership in Hybrid Autonomous Systems Engineering (T-B PHASE)".

References

1. Bashyal S, Venayagamoorthy GK, (2008) Human swarm interaction for radiation source search and localization. In: (2008) IEEE Swarm Intelligence Symposium. SIS 2008: https://doi.org/10.1109/SIS.2008.4668287
2. Becker A, Ertel C, McLurkin J (2014) Crowdsourcing swarm manipulation experiments: a massive online user study with large swarms of simple robots. In: Proceedings—IEEE International Conference on Robotics and Automation pp. 2825–2830. https://doi.org/10.1109/ICRA.2014.6907264
3. Colledanchise M, Ogren P (2017) Behavior trees in robotics and ai: an introduction. CRC Press, Boca Raton
4. Corne D, Reynolds A, Bonabeau E (2008) Swarm Intelligence. Swarm Intell. https://doi.org/10.1007/978-3-540-74089-6
5. Daily M, Cho Y, Martin K, Payton D (2003) World embedded interfaces for human-robot interaction. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, HICSS 2003 pp. 1–6. https://doi.org/10.1109/HICSS.2003.1174285
6. Hsiang TR, Arkin EM, Bender MA, Fekete SP, Mitchell JS (2004) Algorithms for rapidly dispersing robot swarms in unknown environments. In: Algorithmic Foundations of Robotics V pp. 77–93. https://doi.org/10.1007/978-3-540-45058-0_6. arXiv:cs/0212022
7. Jones S, Studley M, Hauert S, Winfield A (2018) Evolving behaviour trees for swarm robotics. Springer Tracts Adv Roboti. https://doi.org/10.1007/978-3-319-73008-0_34
8. Kapellmann-Zafra G, Salomons N, Kolling A, Groß R (2016) Human-robot swarm interaction with limited situational awareness. In: International Conference on Swarm Intelligence, vol. 9882 LNCS, pp. 125–136. Springer. https://doi.org/10.1007/978-3-319-44427-7_11
9. Kira Z, Potter MA (2009) Exerting human control over decentralized robot swarms. In: ICARA 2009—Proceedings of the 4th International Conference on Autonomous Robots and Agents pp. 566–571. https://doi.org/10.1109/ICARA.2000.4803934
10. Kolling A, Nunnally S, Lewis M (2012) Towards human control of robot swarms. HRI'12—Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction pp. 89–96. https://doi.org/10.1145/2157689.2157704
11. Kolling A, Sycara K, Nunnally S, Lewis M (2013) Human swarm interaction: an experimental study of two types of interaction with foraging swarms. J Hum Robot Interact 2(2):104–129
12. Kolling A, Walker P, Chakraborty N, Sycara K, Lewis M (2016) Human interaction with robot swarms: a survey. IEEE Trans Hum Mach Syst 46(1):9–26. https://doi.org/10.1109/THMS.2015.2480801
13. Koza JR (1994) Genetic programming as a means for programming computers by natural selection. Stat Comput 4(2):87–112. https://doi.org/10.1007/BF00175355
14. Lim CU, Baumgarten R, Colton S (2010) Evolving behaviour trees for the commercial game DEFCON. https://doi.org/10.1007/978-3-642-12239-2_11
15. Marjovi A, Marques L, Penders J (2009) Guardians robot swarm exploration and firefighter assistance. In: Workshop on NRS in IEEE/RSJ international conference on Intelligent Robots and Systems (IROS)
16. McLurkin J, Smith J (2008) Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. Distrib Auton Robot Syst 6:399–408. https://doi.org/10.1007/978-4-431-35873-2_39
17. Nagavalli S, Chien SY, Lewis M, Chakraborty N, Sycara K (2015) Bounds of neglect benevolence in input timing for human interaction with robotic swarms. In: Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction, pp. 197–204. https://doi.org/10.1145/2696454.2696470
18. Ogren P (2012) Increasing modularity of UAV control systems using computer game behavior trees. In: AIAA guidance, navigation, and control conference, pp. 1–8. https://doi.org/10.2514/6.2012-4458
19. Saska M, Vonásek V, Chudoba J, Thomas J, Loianno G, Kumar V (2016) Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles. J Intell Robot Syst Theory Appl 84(1–4):469–492. https://doi.org/10.1007/s10846-016-0338-z
20. Walker P, Amraii SA, Lewis M, Chakraborty N, Sycara K (2014) Control of swarms with multiple leader agents. In: Conference proceedings—IEEE international conference on systems, man and cybernetics pp. 3567–3572. https://doi.org/10.1109/smc.2014.6974483
21. Walker P, Nunnally S, Lewis M, Chakraborty N, Sycara K (2013) Levels of automation for human influence of robot swarms. In: Proceedings of the Human Factors and Ergonomics Society pp. 429–433. https://doi.org/10.1177/1541931213571093
22. Walker P, Nunnally S, Lewis M, Kolling A, Chakraborty N, Sycara K (2012) Neglect benevolence in human control of swarms in the presence of latency. In: Conference Proceedings-IEEE International Conference on Systems, Man and Cybernetics pp. 3009–3014. https://doi.org/10.1109/ICSMC.2012.6378253