Data Communication Assistance via Swarm Robotic: A Behaviour Creation Comparison

Phillip Smith[1(✉)], Asad Khan[1], Aldeida Aleti[1], Vincent C. S. Lee[1], and Robert Hunjet[2]

[1] Faculty of Information Technology, Monash University, Clayton, Victoria, Australia
phillip.smith@monash.edu

[2] Defence Science and Technology Group, Edinburgh, South Australia

Abstract. This paper compares two behaviour creation algorithms for Unmanned Aerial Vehicle swarms. The objective of this work is to have a self-organising robotic swarm which unintrusively assists humans in network restricted environments by facilitating data-transfer between disconnected groups. The behaviour creation algorithms explored are a well-cited approach for evolving neural networks (ENN) and our Learning Classifier System (LCS) approach. We utilise simulations in randomised, obstacle dense landscapes to validate, compare and contrast these two algorithms. We explore the algorithms both with and without the environment layout being known to the swarm *a priori*. These simulations show the ENN algorithm struggles to create appropriate behaviours for this complex data-transfer task, resulting in an unassertive swarm. Our LCS behaviours have the swarm operate with up to 61% of the fitness range above ENN in both expected and unexpected environments, resulting in the desired human assistance.

1 Introduction

Swarm robotics is a growing research field which utilises a large collection of low-complexity robotic agents to solve tasks beyond the scope of a single, high-complexity, robot [23,24]. These tasks are achieved by the interaction and cooperation of agents, forming an *emergent behaviour*. One such task for which swarms show potential the creation of a Mobile Ad-hoc NETwork (MANET) [10,12] in environments constricted both spatially and in communication. This swarm task aims to facilitate connectivity between disconnected networking devices in urban environments. More specifically, the swarm assists human operators in network restricted environments to remain in communication with one another, with minimal interaction from the humans or their devices.

In this paper, the autonomous creation of such swarm behaviours is explored via two algorithms: an algorithm which modifies a Neural Network (NN) using a Genetic Algorithm (GA), hereon referred to as Evolved Neural Network (ENN),

A. El Fallah Seghrouchni and D. Sarne (Eds.): IJCAI 2019 Workshops, LNAI 12158, pp. 111–126, 2020.
https://doi.org/10.1007/978-3-030-56150-5_6

and an adaption of Learning Classifier System (LCS) [15] for rule-based evolution and Reinforcement Learning (RL). The former of these approaches has seen considerable exploration within the swarm community [13,14,25]. However, it has only been implemented for trivial swarm tasks, such as locomotion. It is thus hypothesised that ENN will create insufficient behaviours for the challenging data-transfer task, which is non-deterministic, partially-observable and requires heterogeneous agent behaviours. In contrast, the latter approach, LCS, has shown significant performance in our previous data-transfer swarm work [22] and thus is further explored in this study.

The quality of the two algorithms is evaluated via virtual swarms, operating in numerous 2D landscapes with obstacles and restricted communication. Each swarm implementation is tasked with transferring data from one human-held networking device to another in the shortest time possible. This quality-evaluation is measured via task effectiveness, behaviour reliability, and behaviour creation time. These three aspects are seen as key for real-world swarm applications to be operationally viable. Additionally, the quality of the created behaviours is tested in both the environments used by the creation algorithms, and environments unseen *a priori*.

The two main contributions of this paper are:

(a) the implementation and analysis of ENN for swarm behaviours in a non-trivial task
(b) the comparison of ENN and LCS for swarm behaviour creation, in relation to swarm performance, reliability and time efficiency

The remainder of this paper is structured with Sect. 2 presenting a background on swarm ENN and LCS; the agent architecture and modification to the ENN algorithm presented in Sect. 3; Sect. 4 defines the experiments of this study; Sect. 5 presenting the results; and finally this work is concluded in Sect. 6.

2 Related Work

2.1 Swarm ANN

Since early robotic-NN works [3], shallow NNs have been evolved to connect sensor inputs, such as IR range values, with actuator controls, such as individual motor velocities. In these early works, a binary GA was used for autonomous selection of the available sensor values to be used as NN inputs. This search allowed for redundant environment data to be removed from the agents' observation. Furthermore, by evolving the *input activation* genome, the development of an agent was accelerated as this hyper-parameter did not require manual tuning or exhaustive exploration. In contrast to this input evolution, [8,18,25] had respectively: a single, a group and a swarm of robotic agent(s), equipped with fully connected, single-layer NNs. The GAs of these works created behaviours by evolving the neuron weights, connecting the input and output nodes. These agents were evaluated with simplistic tasks, such as moving to a goal location.

These two approaches have been combined in recent swarm evolution studies by Heinerman *et al.* [13,14]. In these works, a swarm was assigned the (simplistic) task of travelling about an environment at maximum speed, without collision. Three-levels of adjustment were utilised to create the required behaviour:

1. The aforementioned evolution of active inputs for the NN was evolved between implementations of the swarm.
2. The heterogeneous evolution of neuron weightings was evolved by each swarm member throughout each trial, with set evaluation windows for measuring weight genome quality.
3. Inter-swarm genome sharing was utilised to accelerate the evolution and reduce behaviour rediscovery [5] in the heterogeneous swarm.

Using multiple evolution layers allowed the swarm to create an overall behaviour genome, which could be stored and reused in later implementations. It also allowed the swarm to be adjustable during operations, overcoming unexpected environment variation. These qualities are also seen in the LCS behaviour creation of [22].

2.2 Learning Classifier System

LCS swarm behaviour creation [22] uses a combination of inter-implementation evolution and intra-implementation RL to create and adjust swarm behaviours created from a set of condition-action (or '*if-then*') rules. These rule-sets are created via the deconstruction of known swarming heuristics, and each rule reconstructed via grammar based genetic programming [19]. During operation, RL has the agents learn appropriate rule use for a given situation. After each swarm implementation, the rule-sets of low performing agents are evolved via the adoption of rules from high performing agents and the mutation of currently held rules.

3 Swarm Task, Agent Architecture and ENN implementation

In this section, the problem domain is further defined and the agent observation methods, available actions and reward function are presented. To improve comparability between ENN and LCS, both utilise the same input variables, action outputs and evaluation functions.

3.1 Problem Domain

In this study, a swarm of Unmanned Aerial Vehicles (UAVs) is simulated assisting the transfer of 125 MB of data between two human-operated devices which are separated by 500 m of random urban landscape and have no satellite connectivity. For simulation simplicity, these devices are abstracted to stationary

communication devices, one being a data sink, the other a data source. Furthermore, this data is discretised into s segments.

For the most effective communication assistance the swarm aims to achieve full data-transfer in minimum time, thus behaviour solutions are evaluated via,

$$\text{fit}_{\boldsymbol{\Gamma}} \in \mathbb{R} : (-1, 1) = \frac{s_s}{s} - \frac{T_s}{T} \tag{1}$$

where s_s is the data-segments that reached the sink within the simulation time-limit, T, and T_s is simulation time. The termination criteria of this task results in either $\frac{s_s}{s} = 1$, causing a positive fitness, or $\frac{T_s}{T} = 1$, causing a negative fitness.

These negative fitnesses are further defined as *shallow failure* and *complete failure*. In the case of shallow failure, all data does not reach the sink devices within the utilised T, but is expected to as $T \to \infty$. In contrast, when fit $\leq \frac{\Phi}{s} - 1$, where Φ is the swarm size, the solution is seen as a complete failure. This fitness suggests each agent only transferred $[0, 1]$ segments during T. Such a score has a close correlation to all agents becoming trapped by obstacles or non-cooperative behaviours emerging. In such situations, the swarm is predicted to fail the data-transfer task even as $T \to \infty$.

3.2 Observation Inputs and Action Outputs

To facilitate data-transfer, agents move toward the goal location with segments in a buffer and transmit these segments to other agents or the non-swarm devices (human communication devices).

An issue that arises with the NN swarms of literature, [14,25], is low-level sensor reading inputs and actuator command outputs cannot realise complex actions without considerable exploration during swarm behaviour evolution. These low-level controls are referred to by Duarte *et al.* in [6] as *primitives*, which are seen as insufficient for 'complex robots in tasks beyond mere locomotion' [6]. As such, a higher level of control is proposed which utilises Low-Level Heuristics (LLH) as both inputs and outputs for the ANN. These LLH are the conditions and actions used in the LCS behaviours, as defined in [22].

For the inputs of the agents, the philosophies of swarm robotics in [4] are upheld by limiting sensor complexity. The inputs are restricted to fellow swarm agent positions, determined via signal triangulation, and network status, determined via information propagation. Agents are also aware of currently observable obstacles by simple range sensors; however, no environment mapping is utilised.

For the action outputs of the swarm agents, five networking and relative motion actions are utilised. These actions are further defined via alternative targeting, giving a total of 15 options. ENN selects from these actions by linking each to an output node; the output node with the highest activation value is executed, similar to [17]. For LCS, each rule associates an environment input condition to one of these actions, creating a discrete pool of possible actions in each rule-set. In this study these actions are:

1) collect from source
2, 3) send to closest neighbour
- toward source
- toward sink

4–7) move toward
- closest neighbour
- source
- sink
- wall

8–11) move away from
- closest neighbour
- source
- sink
- wall

12–15) orbit
- closest neighbour
- source
- sink
- wall

It can be observed from this action list that the swarm agents are responsible for both requesting data from the source and delivering the data to the sink. In this way, the connected devices may remain unaware of the connection being swarm-based; no swarm statuses or behaviour information is fed into the non-swarm devices nor must the non-swarm devices govern the flow of data. As such, the swarm may assist data communication transparently and without requiring the humans or connecting devices to alter their activities to interact with or assist the swarm.

3.3 Reward Function

Swarm agents evaluate local performance via a Geographical routing [11] based reward, ρ. This function is defined as,

$$\begin{aligned} \rho_\alpha &= \sum_{s \in S_h} \Delta(r_{s,\text{sink}}) + \sum_{s \in S_t} \Delta(r_{s,\text{sink}}) + \sum_{s \in S_r} \Delta(r_{s,\text{sink}}) \\ \rho_\beta &= \begin{cases} 0, & |S_h| > 0 \vee |S_t| > 0 \vee |S_r| > 0 \\ \Delta(r_{i,\text{source}}), & \text{otherwise} \end{cases} \\ \rho &= \log(|\rho_\alpha| + 1) \cdot \text{sgn}(\rho_\alpha) + \log(|\rho_\beta| + 1) \cdot \text{sgn}(\rho_\beta) - c \end{aligned} \tag{2}$$

where

- ρ_α is the reward generated by data-segments in S_h, S_t and S_r, which are respectively the agent's collections of data-segments held, transferred and received since last scoring
- ρ_β is the reward a data-less agent receives for moving toward the source
- $\Delta(r_{s,sink})$ is the change in distance between a data segment and sink since last scoring
- $\Delta(r_{i,source})$ is equivalent for agent and source
- c is a cost value for the performed action (communication or movement), which may be weighted to encourage greater agent energy efficiency.

This reward equation encourages agents to effectively transport and transmit data in the direction of the known sink, or return to the source for further data collection.

3.4 Evolved Neural Network Architecture

In alignment with Heinerman [14], the NN structure used by ENN has no hidden layers and thus input and output nodes directly connecting with a weight set W, where the connection of input i and output o is $w_{i,o} \in W, w \in \mathbb{R} : (-1, 1)$. These weights are periodically evolved during the swarm implementation and *input activation* genomes, Γ_{enn}, are evolved offline, between swarm implementations. After empirical exploration, it has been found the third level of adjustment in [14], periodic inter-swarm W exchange, does not results in performance improvement in this swarm implementation. However, the additional offline evolution of *starting weights*, $\bar{W}$, and the use of Γ_{enn} and $\bar{W}$ sharing between swarm members during offline evolution shows some performance improvement. This evolution of starting weights reduces the re-evolution of similar W for each implementation.

During implementation, the current W, W_n, is evaluated via the sum of action rewards, ρ, over period T_w. After T_w, a hill-climbing acceptance function is used to replace the current best W, W_{best}, with W_n if $(\sum \rho)_{best} < (\sum \rho)_n$, or discard W_n otherwise. Should W_n be discarded, the mutation rate, ψ, is modified by a constant increment value, $\dot{\psi}$, via $\psi + \dot{\psi} \rightarrow \psi$. If W_{best} is replaced, ψ it set to $1 \cdot \dot{\psi}$. The potentially updated W_{best} is then mutated for further exploration; $\forall w_{i,o}$ being adjusted via a Gaussian with mean 0 and ψ standard deviation.

After each swarm evaluation is terminated, offline evolution is conducted on the Γ_{enn} and $\bar{W}$. As stated above, this evolution utilises behaviour sharing between swarm members. In this study, this behaviour sharing consists of high-performing swarm members transferring a copy of Γ_{enn} and $\bar{W}$ to low-performing swarm members. Agents' individual performances are measured via the number of data segments that were held by the agent for at least one time-step in the evaluation and reached the sink by implementation end. The threshold between high- and low-performance is the mean of performances over the swarm, as implemented in [21]. Each low-performance agent undergoes offline evolution via crossover and mutation, and a high-performing agent is selected via roulette-wheel to share its behaviour with the evolving agent. The offline evolution of an agent's Γ_{enn} and $\bar{W}$ consists of each genome undergoing one-point crossover. The parents of each crossover are the genome held by the agent and the shared genome. The offspring also undergos mutation via bit-wise flipping for Γ_{enn} and via digit-wise Gaussian shifting for $\bar{W}$. The offline sharing of genomes allows effective individual agent behaviours to be distributed across the swarm, as seen in [14], though the use of this genome in the evolution process allows the swarm to remain heterogeneous, and thus specialised behaviours may still emerge. Additionally, as both the input activation and starting weight evolution are conducted offline, the set of all agents' Γ_{enn} and $\bar{W}$ may be defined as a swarm behaviour solution which remains constant throughout an implementation.

For this data-transfer problem, the structure of the ENN has an input layer of 10 nodes, each with value $\mathbb{R} : [0, 1]$ which are normalised representations of internal and local environment state conditions.

4 Experiment Design

4.1 Simulation

For this study, MASON [16], a Java-based multi-agent simulator, is used for swarm implementation. In these tests, time is discretised into 200 ms time-steps and agents are assumed to travel with mean velocities of $5\,\mathrm{ms}^{-1}$; speeds achievable by UAVs [9]. These time discretisation and agent speeds have been chosen so that 1 time-step represent 1 m of agent travel. Using this discretisation, the file transfer task is broken into 500 data-segments of 2 Mb (0.25 MB) each. Each data-segment is simulated as a single unit, and thus must be wholly transferred within a single time-step. This segment sizing assumes transfer speeds of 10 Mbps, a value within reason for modern Wi-fi [26]. Furthermore, the data-transfer process of these experiments use the Log-Distance Path Loss (LDPL) model [20], which estimates the signal power loss over distance with Gaussian shadowing. In alignment with the communication ability of common Wi-fi adaptors [1], agents have a transfer power of 12 dBm, a receiving threshold of −83 dBm, and a Signal to Noise and Interference Ratio (SNIR) of 10. These signals are transferred at 2.412 GHz (Wi-Fi channel 1 [2]), the environment has a path-loss exponent of 2.5 and Gaussian shadowing with mean 0 and standard deviation 3 dB. The source and sink are placed ∼500 m apart with randomly generated obstacles between. These obstacles attenuate the signal by 5 dB [7] per meter the signal passes through. Finally, the time-limit, T of these tests is 20,000 time-steps, which translates to just over one hour of swarm operation.

4.2 Evolution Parameters

To evolve W in ENN, T_w is set to 20 time-steps and $\dot{\psi}$ to 10^{-3}. For the crossover and mutation of input activation genomes and $\bar{W}$, the mutation rate is set to 5% (to match [14]) and the Gaussian shift of $\bar{W}$ has mean 0 and standard deviation 0.25. Finally, LCS has an evolution generation limit of 500, and ENN has a limit of 1,000. The additional ENN generations allow both algorithms to reach evolution search convergence.

4.3 Experimental Settings

To explore these two algorithms, two experiments are conducted for each: environment-specific evolution and environment-generic evolution.

In the first experiment, 10 sets of 30 environments are presented to LCS and ENN for behaviour evolution. The best-evolved fitness, according to (1), is recorded for each environment and the median and inner quartile of these best fitnesses are reported for each of the ten sets along with the results of a Mann-Whitney U-Test, with $\alpha = 0.05$, to validate the variation in performances. Each of the environments use random generation to determine the obstacle configurations and the initially generated behaviours of the swarm. These generators are uniquely seeded for each of the 30 environments, and the same seed is used

for both LCS and ENN to ensure comparable results. The ten environment sets explore variation in the task settings; validating the comparison between the algorithms. In these sets, the background noise is −85 dBm or −95 dBm and each noise is tested with a swarm of size 1, 2, 4, 8 and 16.

In addition to the fitness results, the percentages of best-evolved solutions which *shallow fail* or *completely fail* the data-transfer task are reported, as task reliability is seen as a key attribute for evaluating the swarm behaviour. Furthermore, the evolutionary improvements of ENN and LCS are reported via the median and inner quartiles of fitness differences between first generations and best-evolved generations for each experiment set.

The second experiment of this study explores the algorithms' abilities to create generally applicable behaviours. A further 30 behaviours are evolved in this experiment with each generation evaluated by using the mean fitness in five training obstacle configurations. The resulting behaviour is tested in a further 30 alternative obstacle configuration environments, giving a total of 900 evaluations per algorithm. In this second experiment, the background noise and swarm size remain constant at −95 dBm and 8, respectively. This experiment explores the generality and reusability of the created behaviours. For both algorithms, the post-evolution behaviour fitnesses and alternative environment fitnesses are presented via median and inner quartiles, and the Mann-Whitney U-Test, fail-rates and evolution progress are again reported. Additionally, the evolution fitness growth of the algorithms is explored via a moving average graph, with a window of 50 generations, and via a graph of the best-evolved solution fitness by each generation. For each algorithm, these graphs are combined for all 30 evolution runs, with the reported values being the average and 90% confidence intervals of the set.

Finally, the processing time required by ENN and LCS to create behaviours are reported. The average time over all evolutions of both above experiments are presented tabularly, along with the average number of time-steps in these evolutions. Additionally, these times and time-step results are graphically presented for further discussion and justification.

5 Results and Discussion

5.1 Environment-Specific Evolution

Shown in Fig. 1 is the median and inner quartiles of both approaches for the discussed background noise and swarm size settings. LCS produces significantly greater fitness solutions over ENN, with all sets showing no quartile overlap. Additionally, the Mann-Whitney U-tests reports values less than 0.001 for all swarm sizes and background noises, this indicates the better performance of LCS is statistically significant. The most significant fitness difference is in the high-noise environments with 8 swarm members; the median fitness difference is 61% of the fitness range ($\frac{0.35--0.872}{1--1}$).

In addition to fitness superiority, Table 1 shows LCS swarms to be more reliable in completing the data-transfer task, with LCS swarms never completely

failing the task. In contrast, ENN swarms see some complete failure cases for most swarm sizes with −85 dBm and for 1-agent swarms with −95 dBm background noise.

Figure 2 shows the median and inner quartile for the evolutionary fitness-improvement. These graphs show the evolution fitness-improvement of both LCS and ENN follow bell-shaped relations to the swarm size, though the algorithms in the two noise settings demonstrate different aspects of this relation. Firstly, ENN demonstrates the lower limit of the relation; for high noise environments, with swarm size ≤ 4, no fitness-improvement is seen. In contrast, ENN swarms of 8 to 16 agents in −85 dBm noise shows a positive relation between swarm size and fitness increase. Likewise, all LCS swarms in −85 dBm noise and all ENN swarms in −95 dBm show a steady increase in fitness-improvement for sizes 1 to 8. An improvement plateau then occurs for these swarms (LCS in −85 dBm and ENN in −95 dBm) when the swarm size increases from 8 to 16. These plateaus have similar fitness-improvements for all swarm sizes and Fig. 1 reports similar or greater final fitnesses. This represents the saturation limit of the fitness-improvement possible by the evolutionary searches; further swarm size increments are predicted to not produce greater fitnesses. Finally, LCS swarms of 8 to 16 agents in −95 dBm noise show a decline in fitness-improvement although Fig. 1 reports these settings both produce high fitness swarms. This suggests the data-transfer task has been simplified by the low noise and large swarm size, and thus requires little (or no) evolution by LCS; the RL of LCS and the variation of rule-sets in the large heterogeneous swarm allows first generation swarms to complete the data-transfer task with significant throughput.

From this pattern observation, LCS leads ENN relative to swarm size impacting fitness-improvement; LCS reaches improvement plateau as ENN begins increasing, and undergoes improvement decline as ENN reaches the plateau. Additionally, it can be observed that both LCS and ENN swarms in −95 dBm, with sizes of 8 and 16, have saturated evolution improvements, yet, Fig. 1 reports

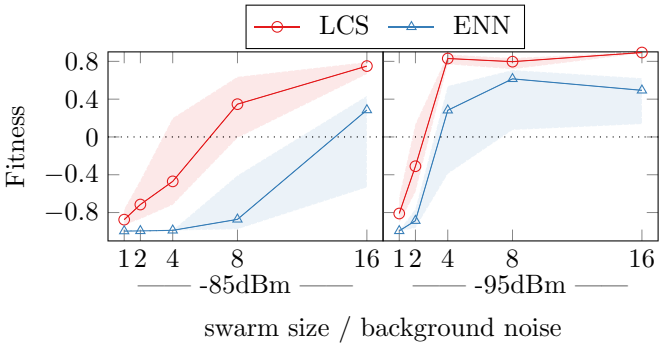

Fig. 1. Median & inner quartiles for each algorithm after evolution. LCS has significantly higher *median*, with no quartile overlap, compared to ENN for all experiment settings.

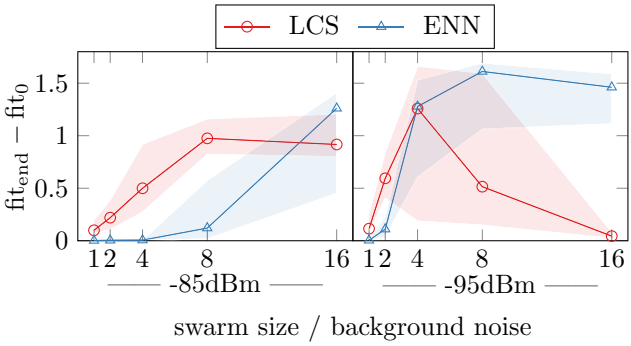

Fig. 2. Median and inner quartile for fitness-improvement during evolution. Both algorithms have little growth when the swarm size is small or when the task is more challenging due to extra background noise.

LCS to produce swarms of greater fitness. Therefore, LCS still dominates ENN with both in optimal conditions. In these optimal settings, LCS has a median fitness of 0.89, which translates to a data-transfer time of 7 min, 20 s or an average transfer of 2.27 Mbps. In contrast, the peak median fitness of ENN is 0.615, which equates to 25 min, 40 s or 650 kbps. It can therefore be concluded, the agent architecture of LCS , particularly the online adjustment of RL rather than periodic weight evolution, allows for overall greater swarm performance.

5.2 Environment-General Evolution

Figure 3 shows the median and inner quartile fitness for the best-evolved behaviours in the training environments, and in the 900 unseen evaluation environments. Additionally, this figure includes evolutionary fitness-improvements.

The LCS swarm shows relatively small reductions in evolution performance when comparing the environment-specific results, Fig. 1, and multi-environment, Fig. 3, results; a median fitness reduction of only 4% is seen. In contrast, the evolution of ENN is significantly impacted by evaluating each generation with

Table 1. Shallow and complete failure rates for environment-specific tests. Fail-rates show LCS never completely fails the task and is less impacted by swarm size than ENN.

		–85 dBm					–5 dBm				
		1	2	4	8	16	1	2	4	8	16
LCS	Shallow failure (%)	100	100	66.7	30	6.7	100	66.7	3.3	6.7	0
	Complete failure (%)	0	0	0	0	0	0	0	0	0	0
ENN	Shallow failure (%)	100	100	100	86.7	46.7	100	100	36.7	20	16.7
	Complete failure (%)	26.7	10	0	10	3.3	13.3	0	0	0	0

multiple environments; the median fitness of ENN reduces by 72.5% of the fitness range ($\frac{0.71-(-0.74)}{1-1}$) and is below fitness 0.

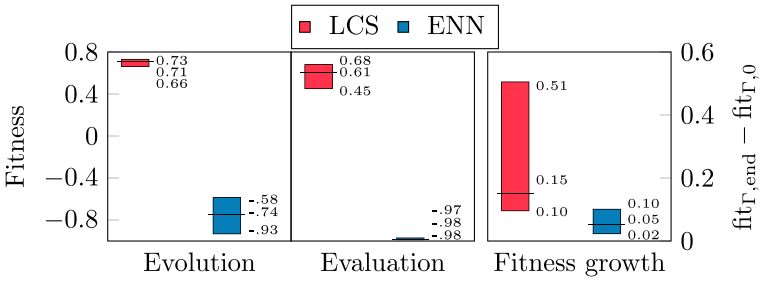

Fig. 3. Median and inner quartile results of best fitnesses during evolution (left), fitness-improvement during evolution (right) and behaviours implemented in unseen environments (centre). Significantly lower evolution progress by ENN results in complete failure in alternative environments.

Furthermore, the created behaviours of LCS prove effectively general with the median fitness of the 900 alternative environments being relatively high at 0.61. This shows even in inexperienced situations the LCS swarm may still transfer data with effective throughput. Additionally, the shallow failures, reported in Table 2, remain relatively low at 13.5% and only 5.2% of cases showed complete data-transfer failure. In comparison, ENN shows considerably lower performance in alternative environments, with all trials being shallow failures, and 36.7% of trials completely failing. As a result of this, a fitness difference of 79.5% arises between ENN and LCS, and LCS produces more reliable solutions. As can be expected from these results, the Mann-Whitney U-tests again reports statistically significant differences, with values less than 0.001. Initially, this poor ENN performance was speculated to be the result of acceptable, but over specialised, behaviours evolving for the five evolution environments and thus not being reusable in the alternative environments. However, the low evolution fitness in Fig. 3 (left) and low evolution improvement in Fig. 3 (right) shows this over specialisation failure is more significant; the ENN evolution cannot createNN which are generally applicable to all five evolution environments, even when given 1000 generations to do so, much less be reusable in the unseen environments. To further explore this failure, Fig. 4 presents the fitness change during evolution for both LCS and ENN via a moving average fitness over the generations, and the best-found fitness during evolution. These graphs show the LCS evolution often finds low-performing solutions, though intermittently discovers high-performing solutions which allow the average maximum fitness to rise to 0.6 by generation 10. In contrast, ENN sees little improvement and no NN configurations are found to solve the task in all evaluation environments. This evolution failure is due to the neural weights requiring specialised settings for each environment. In contrast, the condition-action rules of LCS observe the

Table 2. Shallow and complete failure rates for environment-general tests, both after evolution and when evaluated in alternative environments. Fail-rates show LCS has minor shallow and complete failures, while ENN has over seven times the complete failures and all trials resulted in shallow failure.

		Evolution	Evaluation
LCS	Shallow failure (%)	0	13.2
	Complete failure (%)	0	5.2
ENN	Shallow failure (%)	100	100
	Complete failure (%)	0	36.7

environmental state via hyper-plane dissection, and allow wider application of each rule. Thus the LCS rules has greater applicability during evolution and unseen evaluation.

In relation to transfer speeds in the unseen environments, LCS achieves a median transfer time of 26 min or 641 kbps. Although this value is considerably lower than in environment-specific cases, ENN sees greater reductions with a median of 10 data segments transferred in T, giving an average throughput of 500 bps. It can thus be concluded that ENN is incapable of evolving generally applicable solutions for the application of data-transfer, and thus can only be utilised when an environment is known *a priori*. Additionally, it can be noted that LCS behaviours in unknown environments and ENN in known environments achieve relatively similar performances. This strengthens the argument

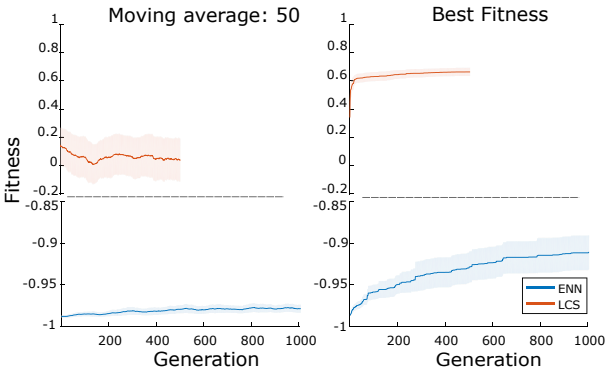

Fig. 4. ENN and LCS average generation fitnesses during environment-general behaviour creation. Left graphs are rolling average fitness with window size of 50 generations. Right graphs are best fitness by generation. Each graph averaged over 30 evolution instances with the averages shown by solid line and 95% confidence intervals shown by shaded areas. LCS has low average fitness but high performing solutions are intermittently discovered. ENN remains low throughout the evolution. As stated in Sect. 4, ENN is evolved for twice the generations of LCS.

for LCS being capable of producing behaviours which are generally applicable in environments unseen before.

5.3 Execution Time

To finalise this study, the average execution time for ENN and LCS are listed in Table 3 for both environment-specific and environment-general behaviour creation. Furthermore, to examine these time differences, the average time-steps are also listed in this table, and the two values are graphically presented in Fig. 5. For both environment-specific and environment-general evolution, ENN requires longer execution time than LCS. For the environment-specific evolution, an execution time 3.7 times that of LCS is observed and when multiple environments are used for evaluation, this multiplier grows to 5.2. This difference is due to two main factors: the lower fitness of ENN throughput evolution requiring more time-steps, and the evolution process of ENN being generally slower than LCS.

In relation to the additional time-steps, a lower fitness directly attributes to longer execution time as each evaluation with a fitness in the range $[-1, 0]$ operates for T time-steps while a solution with positive fitness requires $T_s = T \cdot (1 - Fit)$ steps. In Table 3, ENN is shown to perform, on average, 3.1 and 3.8 times the time-steps of LCS (for specific and general evolution, respectively).

Table 3. Average real-world evolution time (in minutes) and average time-steps for LCS and ENN (multiplied by $T = 20,000$). LCS is more time effective in both experiments.

	Env. Specific		Env. General	
	LCS	ENN	LCS	ENN
Average execution time (min)	60	220	120	620
Average time-steps ($\times T$)	319	988	471	1778

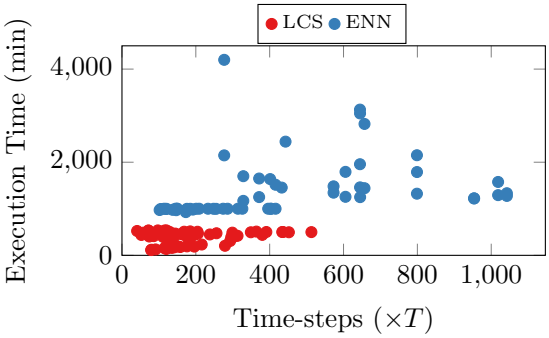

Fig. 5. ENN and LCS graphical depiction of execution time versus time-steps. ENN has have higher execution time, irrespective of time-steps.

After accounting for ENN performing twice the evolution generations of LCS, these time-step ratios are confirmed to contribute to the longer evolution time.

To demonstrate the evolution operations of ENN are generally slower than LCS, Fig. 5 shows the two algorithms hold relatively constant execution times for the time-step range 0 to 300. In this range ENN shows a constant execution time 245% that of LCS. After accounting for the additional evolution generations in ENN, this ratio confirms the evolution and operation of LCS to be generally faster and thus more efficient than ENN.

Table 3 and Fig. 5 shown that LCS is a more efficient behaviour creating algorithm within the context of the data-transfer task of this study. This efficiency is due to both the higher performance throughout evolution allowing faster evaluations and the evolution process for genome alteration being significantly faster.

6 Conclusion and Further Work

In this paper, we have explored two algorithms for creating behaviours which allow a swarm of robots to assist humans in communication restricted, urban environments by facilitating data-transfer between them; a challenging task by swarm robotic standards. Furthermore, these behaviours are required to have the swarm unintrusively interact with the communication devices in a way that the devices (and human operators) do not need to adjust their activities to have the swarm assist.

The two algorithms explored are a well-cited technique for evolving neural networks, ENN, and our own LCS-based algorithm.

This investigation has shown our LCS approach to be significantly superior for this real-world application, in regards to swarm performance and reliability while also creating the behaviours in less time. When the environment conditions are explicitly evolved, we see the LCS swarms achieved an average transfer-rate of 2.27 Mbps in optimal conditions, while the ENN swarm reached only 0.65 Mbps. Additionally, we see ENN has a far higher probability of failing to evolve a valid behaviour for a given environment.

For environments not known *a priori*, ENN shows complete failure to create generally applicable behaviours while LCS sees only minor performance reduction. Finally, it has been shown that LCS evolves these behaviours in far less time, showing it is both more effective and efficient.

From this, our hypothesis is validated that ENN, which is seen as state of the art in swarm literature, is better suited to trivial problems, such as locomotion, while our LCS approach may be applied to this real-world task.

In relation to future work, three areas will be explored for this behaviour creation development: further exploring NN driven swarms, further evaluating the behaviours created by these algorithms, and implementing life-long behaviour learning in the swarm.

Firstly, we aim to further explore the potential of neural network-driven swarms. This exploration will examine more traditional approaches of neural networks, such as offline batch training of NN via expert behaviour data. These

techniques have seen significant success in non-swarm robotic behaviour creation and thus show potential for swarm implementation.

Secondly, this exploration has evaluated the algorithms in resulting swarm fitness (which includes task efficiency), reliability and behaviour creation time. Additionally, by evaluating the swarms in conditions unseen during training, the reusability of the created behaviours was assessed. However, for real-world swarm implementations, it is acknowledged that further evaluation work is required. This includes evaluating the energy efficiency of the behaviours, the emergent behaviours' robustness to member loss and mid-operation condition changes, and assessing the predictability of the agents to either improve robot-human cooperation or reduce the swarm's venerability to malicious attack.

Finally, this study has revealed that although LCS creates environment-general behaviours with greater performance than ENN, some performance reduction is still present compared to environment-specific behaviours. To overcome such reduction, our final future work will explore life-long learning in the form of adding each evolved behaviour to a repertoire. During deployment, agents may select from this repertoire in place of evolving new solutions. In doing so, we aim to see the performance of environment-specific behaviours with the wide applicability of environment-general behaviours.

Acknowledgement. Funding for this research was provided by Cyber and Electronic Warfare Division, Defence Science and Technology Group, Commonwealth of Australia.

References

1. RN-XV Data Sheet. https://cdn.sparkfun.com/datasheets/Wireless/WiFi/WiFly-RN-XV-DS.pdf
2. IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements - Part 11: wireless LAN medium access control (mac) and physical layer (phy) specifications. IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012), pp. 1–3534, December 2016. https://doi.org/10.1109/IEEESTD.2016.7786995
3. Balakrishnan, K., Honavar, V.: On sensor evolution in robotics. In: Conference on Genetic Programming, pp. 455–460. MIT Press (1996)
4. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intell. **7**(1), 1–41 (2013)
5. D'Ambrosio, D.B., Lehman, J., Risi, S., Stanley, K.O.: Evolving policy geometry for scalable multiagent learning. In: International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 731–738 (2010)
6. Duarte, M., Gomes, J., Oliveira, S.M., Christensen, A.L.: Evolution of repertoire-based control for robots with complex locomotor systems. IEEE Trans. Evol. Comput. **22**(2), 314–328 (2018)
7. Faria, D.B.: Modeling signal attenuation in IEEE 802.11 wireless LANs. Computer Science Department, Stanford University 1 (2005)
8. Floreano, D., Mondada, F.: Automatic creation of an autonomous agent: genetic evolution of a neural network driven robot. In: From Animals to Animats 3: International Conference on Simulation of Adaptive Behavior, pp. 421–430. The MIT Press (1994)

9. Fotouhi, A., Ding, M., Hassan, M.: Dronecells: Improving 5g spectral efficiency using drone-mounted flying base stations. arXiv preprint arXiv:1707.02041 (2017)
10. Fraser, B., Szabo, C., Hunjet, R.: Simulating the effect of degraded wireless communications on emergent behaviour. In: Proceedings of the 2017 Winter Simulation Conference (2017)
11. Ghafoor, K.Z., Lloret, J., Sadiq, A.S., Mohammed, M.A.: Improved geographical routing in vehicular ad hoc networks. Wireless Pers. Commun. **80**(2), 785–804 (2014). https://doi.org/10.1007/s11277-014-2041-3
12. Hauert, S., Zufferey, J.C., Floreano, D.: Evolved swarming without positioning information: an application in aerial communication relay. Auton. Robots **26**(1), 21–32 (2009)
13. Heinerman, J., Drupsteen, D., Eiben, A.E.: Three-fold adaptivity in groups of robots: the effect of social learning. In: Conference on Genetic and Evolutionary Computation, pp. 177–183. ACM (2015)
14. Heinerman, J., Rango, M., Eiben, A.E.: Evolution, individual learning, and social learning in a swarm of real robots. In: Symposium Series on Computational Intelligence, pp. 1055–1062. IEEE (2015)
15. Holland, J.H., Reitman, J.S.: Cognitive systems based on adaptive algorithms. ACM Sigart. Bull. **63**, 49–49 (1977)
16. Computer Science & Engineering Laboratory: The GMU Center for Social Complexity: Mason. http://cs.gmu.edu/~eclab/projects/mason/
17. Liu, C., Zhu, E., Zhang, Q., Wei, X.: Modeling of agent cognition in extensive games via artificial neural networks. IEEE Trans. Neural Networks Learn. Syst. **99**, 1–12 (2018)
18. Nelson, A.L., Grant, E., Henderson, T.C.: Evolution of neural controllers for competitive game playing with teams of mobile robots. Robot. Auton. Syst. **46**(3), 135–150 (2004)
19. O'Neill, M.: Grammatical evolution: evolutionary automatic programming in an arbitrary language. Kluwer Academic Publishers, Boston (2003)
20. Rappaport, T.S., et al.: Wireless communications: principles and practice, vol. 2. Prentice Hall, PTR (1996)
21. Smith, P., Hunjet, R., Aleti, A., Barca, J.C., et al.: Data transfer via UAV swarm behaviours: Rule generation, evolution and learning. Aust. J. Telecommun. Dig. Econ. **6**(2), 35 (2018)
22. Smith, P., Hunjet, R., Khan, A.: Swarm learning in restricted environments: an examination of semi-stochastic action selection. In: International Conference on Control, Automation, Robotics and Vision, pp. 848–855. IEEE (2018)
23. Soleymani, T., Trianni, V., Bonani, M., Mondada, F., Dorigo, M.: Bio-inspired construction with mobile robots and compliant pockets. Robot. Auton. Syst. **74**, 340–350 (2015)
24. Timmis, J., Ismail, A.R., Bjerknes, J.D., Winfield, A.F.: An immune-inspired swarm aggregation algorithm for self-healing swarm robotic systems. Biosystems **146**, 60–76 (2016). https://doi.org/10.1016/j.biosystems.2016.04.001, https://www.ncbi.nlm.nih.gov/pubmed/27178784
25. Trianni, V.: Evolutionary swarm robotics: evolving self-organising behaviours in groups of autonomous robots, vol. 108. Springer (2008)
26. Vassis, D., Kormentzas, G., Rouskas, A., Maglogiannis, I.: The IEEE 802.11 g standard for high data rate WLANs. IEEE network **19**(3), 21–26 (2005)