Autonomously Learned Behaviour Selection Results

Henry Ye

February 2022

The aim of this project was to implement Environment Specific Behaviour Selection to the already established Multi-Modal Flocking Swarm using an Autonomous Learning approach. The Multi-Model swarm was built using Python's Mesa simulation library and consisted of agents with four modes: FLOCK, AVOID, SEEK and JOIN. FLOCK mode can be seen as the 'default' flight mode, while AVOID mode is entered when an agent closely encounters another swarm agent or obstacle. SEEK mode is activated when an agent is at risk of being separated from the swarm while JOIN mode is activated when an agent is fully separated from the swarm. Environment Specific Behaviour Selection was inspired by previous work done with Hierarchical Graph Neurons which allowed for agents to switch behaviours according to the environment they were in based on pattern matching.

1 Agent Features

The first step in developing an Autonomously Learned Behaviour Selection system was to collect agent features which may be a good indicator of what mode the agent should switch to. A great variety of agent features were initially collected and tested. The main features collected were the following:

- Connectivity
- Number of connections
- Smallest distance from another swarm agent
- Largest distance from another swarm agent
- Smallest distance from nearest obstacle
- Acceleration
- Speed
- Angle

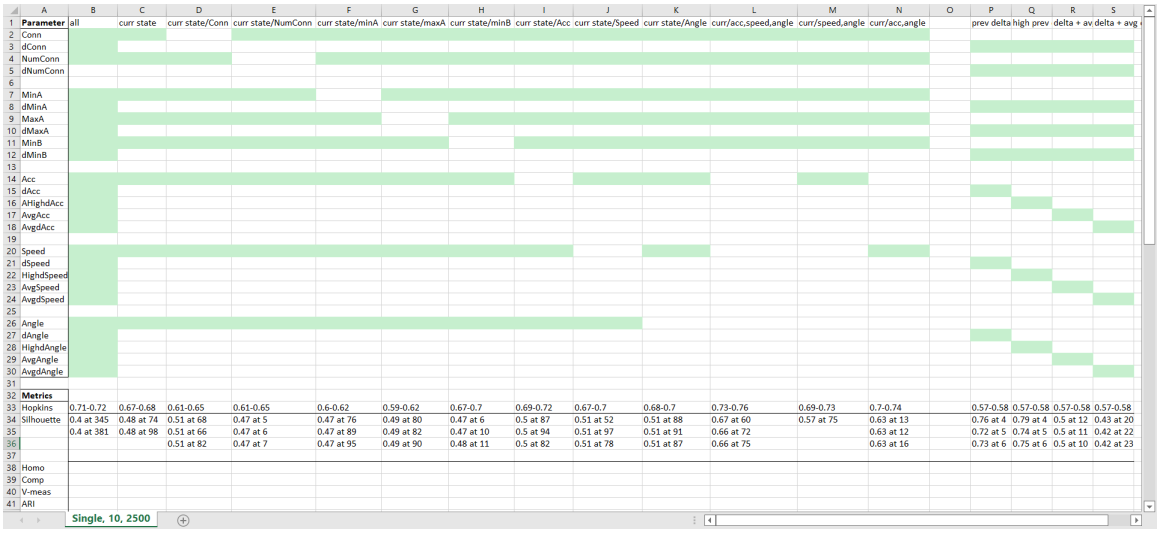
However, there were also different versions of these features that were collected:

- Current
- Previous

- Change in
- Change in change in
- Average in last 10/50/100 steps
- Largest change in the last 10/50/100 steps
- Average change in last 10/50/100 steps

Therefore, there was a vast number of combinations which could be tried due to the many permutations of features. However, some combinations were not necessary to test, as they provide the same information. For example, making a combination of features from the same class e.g. current speed, previous speed, change in speed, etc. Additionally, some different features classes roughly tell the same story e.g. connectivity and number of connections.

Before selecting and trying features, the amount of data or how many steps of the simulation to be saved should be decided on. When debugging the simulation it was best to run the simulation for a smaller number of steps such as 200-500 in order to see some changes in the simulation but not run for too long to minimise time wasted. When properly collecting data however, 1000-1500 steps of the simulation was saved to collect more changes within the environment and to allow the agents to explore more. Ultimately, 1000 steps was landed on since it ends the simulation right before the swarm reaches the other side of the arena and wraps around again.



Parameter	all	curr state	curr state/Conn	curr state/NumConn	curr state/minA	curr state/maxA	curr state/minB	curr state/Acc	curr state/Speed	curr state/Angle	curr/acc,speed,angle	curr/speed,angle	curr/acc,angle		prev delta	high prev	delta + av	delta + avg
Conn																		
dConn																		
NumConn																		
dNumConn																		
MinA																		
dMinA																		
MaxA																		
dMaxA																		
MinB																		
dMinB																		
Acc																		
dAcc																		
AHighdAcc																		
AvgAcc																		
AvgdAcc																		
Speed																		
dSpeed																		
HighdSpeed																		
AvgSpeed																		
AvgdSpeed																		
Angle																		
dAngle																		
HighdAngle																		
AvgAngle																		
AvgdAngle																		
Metrics																		
Hopkins	0.71-0.72	0.67-0.68	0.61-0.65	0.61-0.65	0.6-0.62	0.59-0.62	0.67-0.7	0.69-0.72	0.67-0.7	0.68-0.7	0.73-0.76	0.69-0.73	0.7-0.74		0.57-0.58	0.57-0.58	0.57-0.58	0.57-0.58
Silhouette	0.4 at 345	0.48 at 74	0.51 at 68	0.47 at 5	0.47 at 76	0.49 at 80	0.47 at 6	0.5 at 87	0.51 at 52	0.51 at 88	0.67 at 60	0.57 at 75	0.63 at 13		0.76 at 4	0.79 at 4	0.5 at 12	0.43 at 20
	0.4 at 381	0.48 at 98	0.51 at 66	0.47 at 6	0.47 at 89	0.49 at 82	0.47 at 10	0.5 at 94	0.51 at 97	0.51 at 91	0.66 at 72		0.63 at 12		0.72 at 5	0.74 at 5	0.5 at 11	0.42 at 22
			0.51 at 82	0.47 at 7	0.47 at 95	0.49 at 90	0.48 at 11	0.5 at 82	0.51 at 78	0.51 at 87	0.66 at 75		0.63 at 16		0.73 at 6	0.75 at 6	0.5 at 10	0.42 at 23
Homo																		
Comp																		
V-meas																		
ARI																		

Figure 1: Initial manual feature testing and analysis records in Excel. Records can be found in manualFeatureComp.csv within the 'collect_data' branch.

The initial steps in feature selection involved manually choosing a combination of roughly 5-6 and then calculating their Hopkin's statistic and silhouette score. Hopkin's statistic provides a score based on how well the data is going to cluster, while the silhouette score gives a score for clusterability at a range of different clusters. This obviously proved to be inefficient and sluggish, resulting in the transition to automatic feature selection.

Roughly 50 features were fed through into several feature selection tools at a time provided by the sci-kit learn library. It became apparent early on that increasing the number of features did not correlate with better describing agent mode. Recursive feature elimination was attempted to find the optimal number of features, but the result was rather disappointing. The algorithm often output values greater than 10, less than 3 but generally 5-6 based on the features and number of

features input. Feature selection on the other hand was more promising. Figures 2 and 3 shows the results from Tree-based Feature Selection and Model-based Feature Selection respectively. These two tools performed well and gave a visual representation of feature importance.

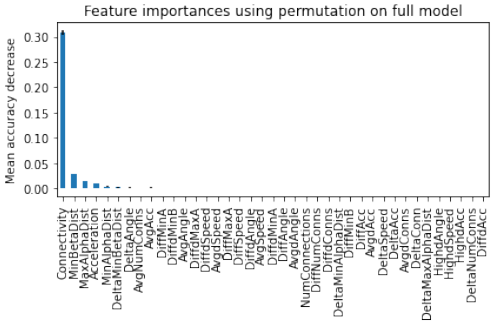


Figure 2: Tree-based feature selection. To reproduce, run 'Feature Importance based on Feature Permutation' cells under 'Tree-based Feature Selection' within featureSelection.ipynb in either 'collect_data' or 'final_version' branches.

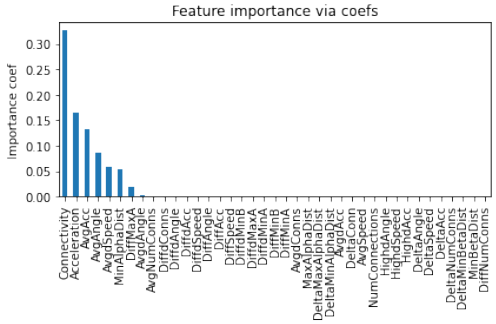


Figure 3: Model-based Feature Selection. To reproduce, run 'Model-based Feature Selection' cells within featureSelection.ipynb in either 'collect_data' or 'final_version' branches.

Forward and Backward feature selection also provided a different perspective on the dataset and were also the two selection tools with the longest run times. All tools had similarities such as noting the importance of the connectivity feature as well as some sort of distance measurement. Some form of acceleration was also found highly ranked in all three of these selection tools. There were also several differences. Tree-based feature selection only deemed current state features important while model-based model selection saw some value in both previous and current states. Conversely, forward and backward selection saw previous state features dominating current state features.

```
Features selected by Forward Sequential Selection: ['MaxAlphaDist', 'DeltaAcc',
'HighdSpeed', 'AvgAcc', 'AvgAngle', 'AvgdSpeed']
Done in 73.732s
Features selected by Backward Sequential Selection: ['MinAlphaDist', 'Acceleration',
'AvgAcc', 'AvgAngle', 'DiffMinB', 'DiffdMinA']
Done in 829.411s
```

Figure 4: Most important features from Forward and Backward Sequential Feature Selection. To reproduce, run 'Sequential Feature Selection' cells within featureSelection.ipynb in either 'collect_data' or 'final_version' branches.

In addition to these more rigorous tools, simpler approaches such as the correlation matrix was also turned to.

	Mode	Found	DangerClose	ConnBelowThresh	Connectivity	DeltaConn	NumConnections	DeltaNumConns	MinAlphaDist	DeltaMinAlphaDist	MaxAlphaDist	DeltaMaxAlphaDist
Mode	1.000	-0.266	-0.107	0.351	-0.295	-0.022	-0.281	-0.019	0.435	0.014	0.462	0.005
Found	-0.266	1.000	-0.075	-0.082	0.052	-0.017	0.065	-0.007	-0.242	-0.016	-0.256	-0.017
DangerClose	-0.107	-0.075	1.000	0.306	-0.259	0.005	-0.238	0.010	0.141	-0.031	0.133	0.007
ConnBelowThresh	0.351	-0.082	0.306	1.000	-0.787	-0.051	-0.781	-0.037	0.541	-0.009	0.451	-0.019
Connectivity	-0.295	0.052	-0.259	-0.787	1.000	0.077	0.927	0.044	-0.579	0.002	-0.480	0.014
DeltaConn	-0.022	-0.017	0.005	-0.051	0.077	1.000	0.070	0.747	-0.039	-0.290	-0.012	0.176
NumConnections	-0.281	0.065	-0.238	-0.781	0.927	0.070	1.000	0.078	-0.670	-0.027	-0.565	0.015
DeltaNumConns	-0.019	-0.007	0.010	-0.037	0.044	0.747	0.078	1.000	-0.050	-0.543	-0.022	0.153
MinAlphaDist	0.435	-0.242	0.141	0.541	-0.579	-0.039	-0.670	-0.050	1.000	0.073	0.974	0.040
DeltaMinAlphaDist	0.014	-0.016	-0.031	-0.009	0.002	-0.290	-0.027	-0.543	0.073	1.000	0.075	0.418
MaxAlphaDist	0.462	-0.256	0.133	0.451	-0.480	-0.012	-0.565	-0.022	0.974	0.075	1.000	0.070
DeltaMaxAlphaDist	0.005	-0.017	0.007	-0.019	0.014	0.176	0.015	0.153	0.040	0.418	0.070	1.000
MinBetaDist	0.031	0.130	-0.400	-0.268	0.224	-0.005	0.207	-0.003	-0.171	-0.004	-0.146	0.004
DeltaMinBetaDist	-0.027	-0.003	0.115	-0.006	0.006	-0.003	0.005	0.003	-0.002	-0.009	-0.003	0.005
Acceleration	0.423	-0.045	0.081	0.373	-0.305	-0.019	-0.286	-0.012	0.277	0.002	0.315	0.037
DeltaAcc	0.004	-0.002	0.036	0.005	0.009	0.040	0.012	0.078	-0.013	-0.122	-0.011	0.088
Speed	0.188	-0.107	-0.274	0.310	-0.329	-0.005	-0.348	-0.006	0.393	0.008	0.358	-0.036
DeltaSpeed	0.042	0.005	-0.014	0.035	-0.018	-0.010	-0.019	0.002	0.021	-0.033	0.029	0.005
Angle	0.172	-0.082	0.175	0.136	-0.116	-0.027	-0.104	-0.019	0.107	0.007	0.126	0.015
DeltaAngle	0.078	0.014	0.092	0.006	0.007	-0.005	0.017	0.010	-0.024	-0.022	-0.008	0.039
HighdAcc	0.001	0.009	0.039	0.017	-0.005	0.043	0.000	0.083	-0.013	-0.129	-0.012	0.091
HighdSpeed	0.034	0.003	-0.008	0.036	-0.017	-0.008	-0.020	0.004	0.023	-0.033	0.031	0.007
HighdAngle	0.051	0.010	0.053	0.003	0.003	-0.010	0.007	0.002	-0.008	-0.008	0.002	0.023

Figure 5: Snippet of feature correlation matrix. To reproduce, run the 'Correlation Matrix' cells within stepRun.ipynb in either 'collect_data' or 'final_version' branches.

A correlation matrix was used to get an idea of which features best correlated with the swarm agent modes, as well as the hard condition behaviour switch flags 'Found', 'DangerClose' and 'ConnBelowThresh'. The combined results from all these tools derived the final combination of features that continued onto the next steps of the method:

- Current connectivity
- Current smallest distance from another swarm agent
- Current largest distance from another swarm agent
- Current smallest distance from nearest obstacle
- Current acceleration

It was surprising that all the final features were based on current state. Previously, it was believed that knowing which mode the agents need to be in required the knowledge of previous states of the agent, but this may not be the case. Logically, this combination of features makes sense, as you have connectivity to provide information on how well connected each swarm member is and whether it should be in FLOCK, SEEK or JOIN mode. The swarm agent distance measurements also reaffirm connectivity for modes such as SEEK, where the agent is at risk of separation and JOIN, where the agent is separated. While distance from obstacles is obviously used to determine whether agents should be in AVOID mode, acceleration is more nuanced. There are some rough ranges and indicators given by acceleration, such as a decrease when in AVOID mode, no acceleration in FLOCK mode, an increase in SEEK and an even greater increase in JOIN mode.

2 Clustering

While the correlation based selection of features (with affirmation from other selection tools) showed promise, the sets of features which were completely output by the feature selection tools were still tried.

So, the combination of features output by model based feature selection (important coefficient) were tried as an experiment. This was the following combination:

- Current connectivity
- Current smallest distance from another swarm agent
- Average acceleration from last 100 steps
- Average angle from last 100 steps
- Average change in speed in last 100 steps
- Current acceleration
- Difference in largest distance from other swarm agents in last 100 steps

Before doing any clustering, Principal Component Analysis was conducted on the dataset in order to reduce the dimensionality of the data and extract the top Principal Components (PC) (PC1, PC2, PC3, etc.). These PCs were then plotted against each other in order to visualise the clustering performance on the principal bases. All following clustering plots are PC1 vs PC2, however later down the track when finalising clustering results, combinations of PC1 vs PC3 and PC2 vs PC3 were also plotted to verify clustering performance in other bases.

The original plan was to simply cluster the dataset into the four modes, but this proved to be difficult. Therefore, the idea was adapted to increase the number of clusters up to 20, and then map the 20 clusters back to the 4 modes later on. Figure 7 is an example of clustering performed on a swarm wide dataset, where the selected features of all swarm agents was included. BIRCH clustering stood out early on as one of the better clustering algorithms, as it was able to cluster some of the complex structures of the data.

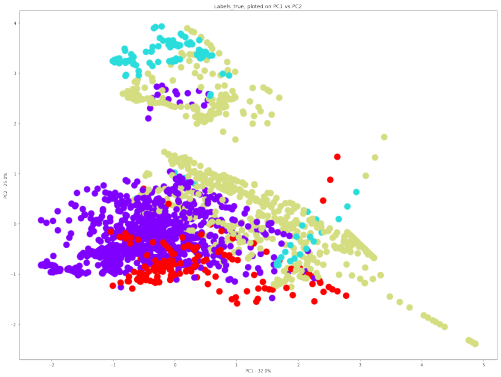


Figure 6: True labelled modes of swarm wide data from 1500 step simulation.

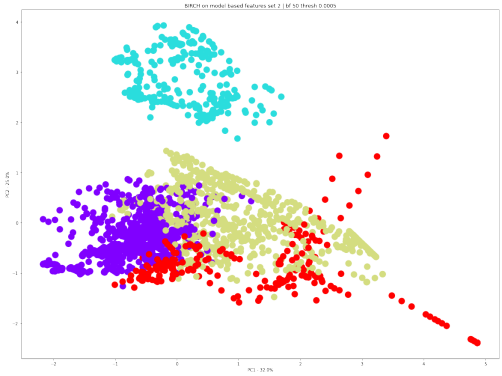


Figure 7: BIRCH clustering into 4 clusters of swarm wide data from 1500 step simulation.

Although this was considered a relatively 'strong' clustering, it still revealed some issues with clustering into just 4 modes, especially since this was still a small dataset. Therefore, increasing the number of clusters may allow the algorithms to match the data's complexity. Let's take a step back and have a look at just a single agent features.

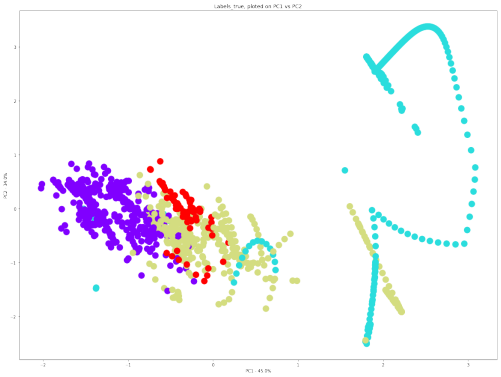


Figure 8: True labelled modes of single agent data from 1500 step simulation.

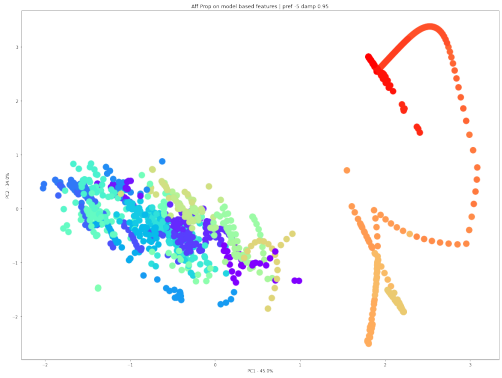


Figure 9: Affinity Propagation clustering into 10 clusters of single agent data from 1500 step simulation.

The following plots were created using a correlation based combinations of features.

Another standout clustering algorithm in the beginning was Affinity Propagation. Although it didn't perform as well as BIRCH in clustering 4 clusters, it could also distinguish some of the data's complex structures as seen in Figure 9. Figure 11 shows its performance for swarm wide data.

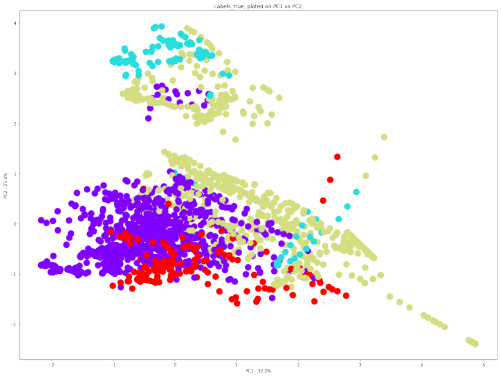


Figure 10: True labelled modes of swarm wide data from 1500 step simulation.

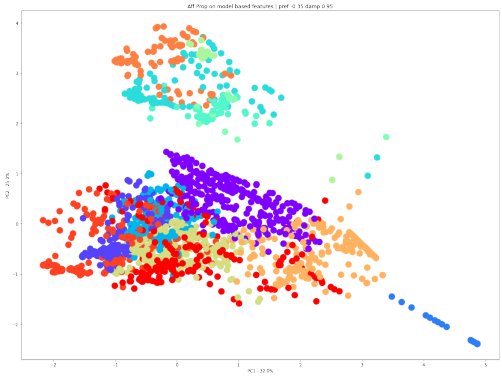


Figure 11: Affinity Propagation clustering into 10 clusters of swarm wide data from 1500 step simulation.

Unfortunately, the downside of Affinity Propagation is that its run-time scales poorly with dataset size. This was not apparent when clustering just single agent data, but even the swarm wide data in Figure 11 took 40-60 minutes to complete. Although the clustering step is only run once, it is still not ideal to have such a long run-time especially when using even larger datasets.

Fortunately, another clustering method which performed well and fast was discovered, Gaussian Mixture Model.

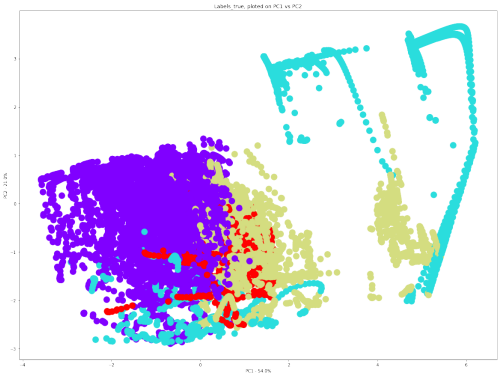


Figure 12: True labelled modes of combined swarm wide data from 1500 step simulation.

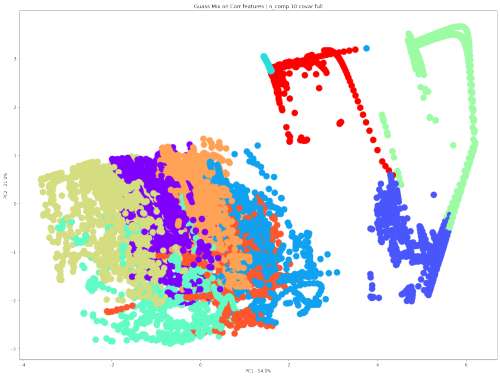


Figure 13: Gaussian Mixture Model clustering into 10 clusters of combined swarm wide data from 1500 step simulation.

After 6 more iterations of correlation based feature combinations, the final clustering result were found. Both BIRCH and GMM showed great results with larger sets of data and were ultimately the only clustering algorithms used in later sets of features.

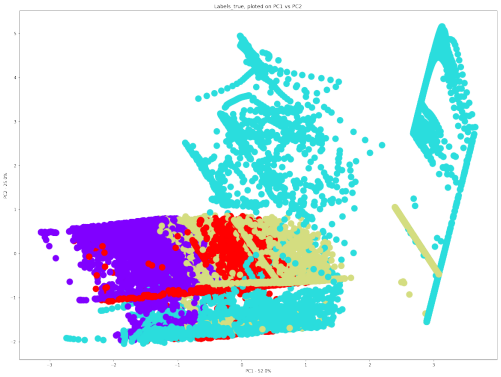


Figure 14: True labelled modes of combined swarm wide data from 1500 step simulation.

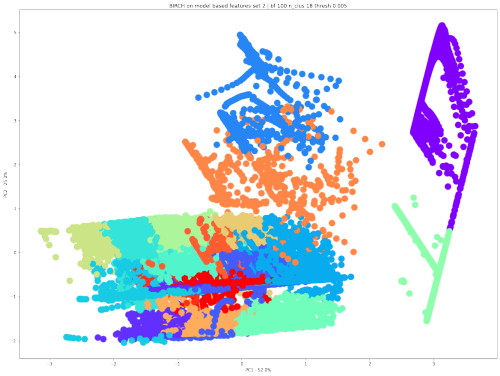


Figure 15: BIRCH clustering into 18 clusters of combined swarm wide data from 1500 step simulation.

The final clustering was performed on combined swarm wide data, where data from multiple different simulations in different environments was joined together to cover a larger diversity of environments and path difficulties. This will ensure that the learned behaviour selection is generalised and not specific to just one or two environments. The swarm size and number of obstacles however was kept constant.

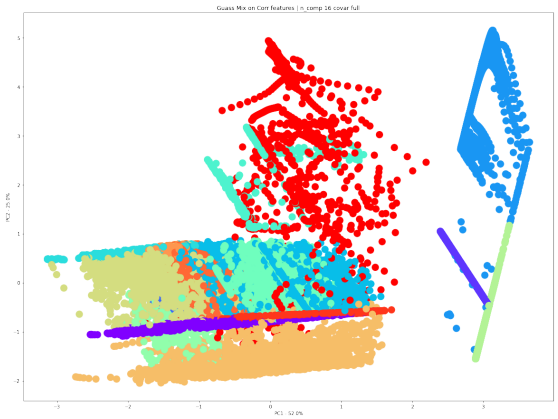


Figure 16: GMM clustering into 16 clusters of combined swarm wide data from 1500 step simulation.

3 Genetic Algorithm

A Genetic Algorithm (GA) was used as an efficient way to search the solution space of the problem and find the best mapping of clusters to modes. These were the methods the GA used:

- Two-point crossover (although initially Single-point crossover was used).
- Uniform integer mutation with a probability of 5%.
- Roulette selection to select parents to mate.
- Cull based on worst fitness.

The Genetic Algorithm was mostly initialised with these hyperparameters:

- Population size: 100
- Simulation steps: 1000
- Generation cap: 200
- Proportion of population selected to mate and cull: 10%

However, when testing and debugging the GA, hyperparameters were drastically decreased to minimise waiting time. This would be an example set of a test/debugging run:

- Population size: 5
- Simulation steps: 200
- Generation cap: 5
- Proportion of population selected to mate and cull: 40%

Note that proportion selected to mate and cull has increased only because population size is so small, and that there needs to be at least 2 individuals selected in order to mate and produce new individuals. Increasing hyperparameters was also tested, such as increasing population size to

300 and generation cap to 1000, but this seemed to be counterproductive as solutions would converge early on anyway.

It can be seen that most mapping solutions in the initial population have high fitness with some (usually just 1 or 2) solutions with already relatively low fitness. This suggests that most of the solutions space is useless and therefore the GA approach skipped trying such solutions. The fact that the initial population already has some solutions with low fitness suggests that there are a plethora of local minima. Eventually, over 10s and 100s of generations, the population converges to the local minima best solution. Generally, around 100-200 generations was a good cap for the problem.

Around 20,000 solutions were collected before the completion of the project and while there have been some standout solutions, best solutions might still be undiscovered due to the sheer size of the solutions space. Here are the best solutions that discovered so far:

1. (2, 4, 4, 4, 4, 1, 3, 4, 4, 3, 1, 3, 1, 1, 3, 4)
2. (1, 1, 1, 3, 3, 3, 1, 3, 1, 3, 1, 3, 3, 2, 3, 4)
3. (3, 2, 4, 1, 3, 3, 3, 2, 4, 3, 2, 4, 3, 4, 3, 2)

Of note, the list position represents the cluster number, while the value in the position represents the mode that that cluster maps to. For example, for standout solution 1, its cluster 1 (position 1) maps to mode 2 which is AVOID.

It was very interesting to see the variety of solutions that the GA output. They had all evolved to excel in different aspects based on the specified fitness equation:

$$fitness = \left(w_{colls} * \frac{m_{colls}}{n}\right) + \left(w_{subfs} * \frac{m_{subfs}}{n}\right) + \left(w_{frags} * \frac{m_{frags}}{n}\right) + (w_{dist} * m_{dist})$$

where w_i is the weighting factor of component i, m_i is the mean of component i, n is the swarm size (10), *colls* is the swarm collisions, *subfs* is the swarm subflocks, *frags* is the swarm fragmentation score and *dist* is the swarm's smallest distance from virtual leader.

A large part of tuning the GA was tuning the weights of the fitness components. Saving all GA solutions to a pickle file which acted like a historical record of all previously attempted solutions. This allowed for post-evolution adjusting of the fitness component weights while still being able to compare past and future solutions. Therefore, the fitness calculation started with all component weights equal to 1. However, this was quickly adapted, as the distance component can usually be several factors larger than the other components. Therefore, the distance weight was greatly reduced to 0.2. The fragmentation component weight was also decreased to 0.5 for the same reason. Therefore, the final weights for the fitness components were $w_{colls} = 1$, $w_{subfs} = 1$, $w_{frags} = 0.5$ and $w_{dist} = 0.2$.

The following are fitness plots from just two of the successful evolutions.

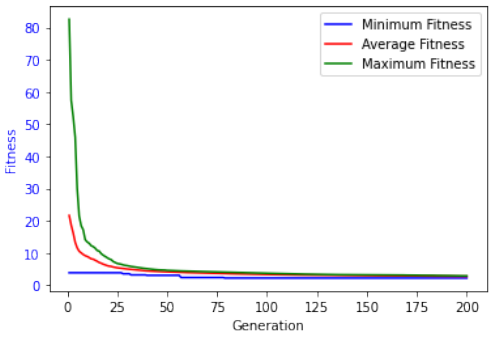


Figure 17: Fitness plot of evolution with 100 population, 1000 simulation steps, 200 generations and 10% mate and replace. To reproduce, run genAlgTest.ipynb in the 'collect_data' branch or runGeneticAlgorithm.ipynb in the 'final_version' branch with the appropriate hyperparameters. Fitness plots will be output every 5 generations as well as at the end of successful evolution.

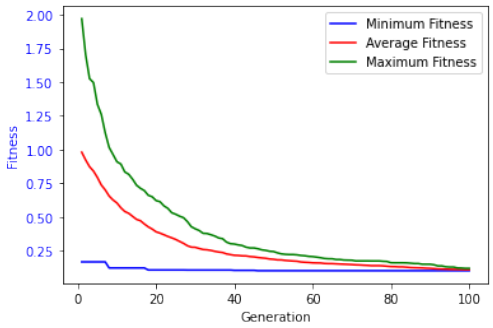


Figure 18: An earlier fitness plot of evolution with 20 population, 1000 simulation steps, 100 generations and 10% mate and replace. To reproduce, run genAlgTest.ipynb in the 'collect_data' branch or runGeneticAlgorithm.ipynb in the 'final_version' branch with the appropriate hyperparameters. Fitness plots will be output every 5 generations as well as at the end of successful evolution.

Once evolution had completed and provided several 'best' solutions, they were ran in the Mesa visualisation first. It was interesting that some of the solutions which had better fitness did not actually perform that well in the visualisation. This may be because they over prioritised in some metric and less in others. This was most noticeable in solutions which achieved a very low score in distance from virtual leader.

After filtering out the solutions which did not actually perform as well as their fitness suggested, it was time to run the standout solutions against the original hard condition behaviour selection. This was done by running the solution as well as original implementation in 50 randomly seeded environments to reduce as much randomness as possible. The performance in the previously mentioned metrics was plotted as violin plots.

Dataset for Figure 19: `https://gitlab.com/phillip.smith/FlockingProject/-/blob/collect_data/Final-Comparisons/Hybrid/comparison_metrics/metric_comparison.csv`.

Dataset for Figure 20: `https://gitlab.com/phillip.smith/FlockingProject/-/blob/collect_data/Final-Comparisons/Hybrid/comparison_metrics/metric_comparison_3.csv`.

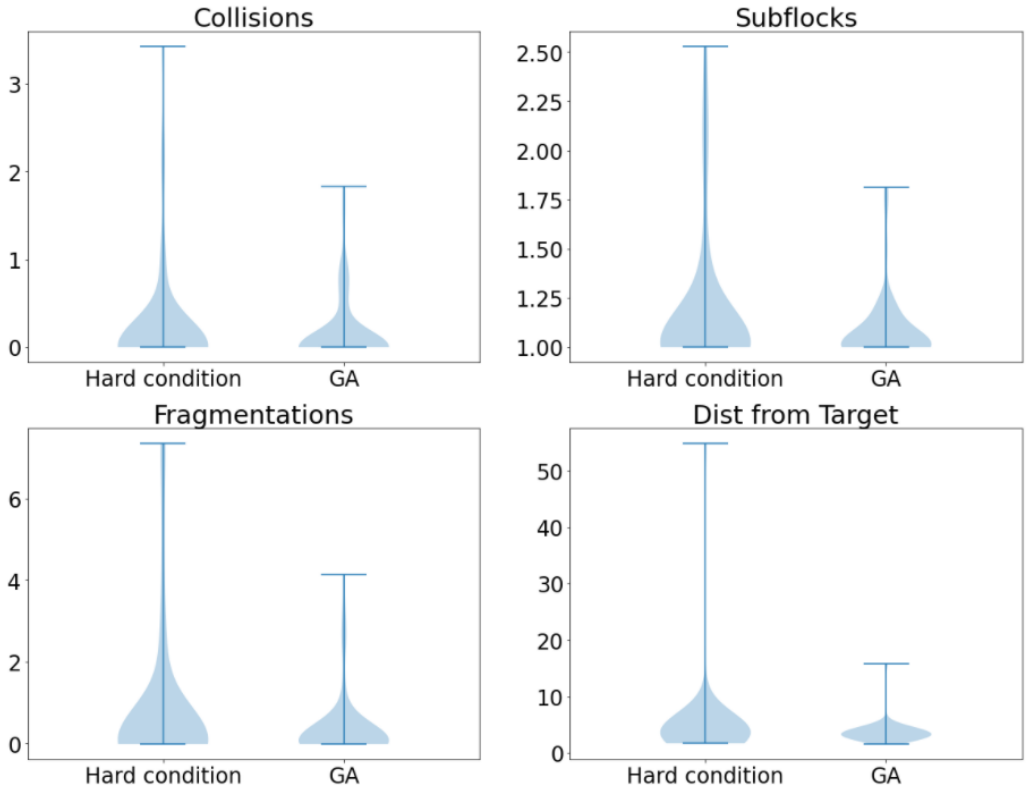


Figure 19: Violin plot comparing metrics of hard condition implementation and GA standout solution 1. To reproduce, run comparisonRuns.ipynb in 'final_version' branch with appropriate list of seeds.

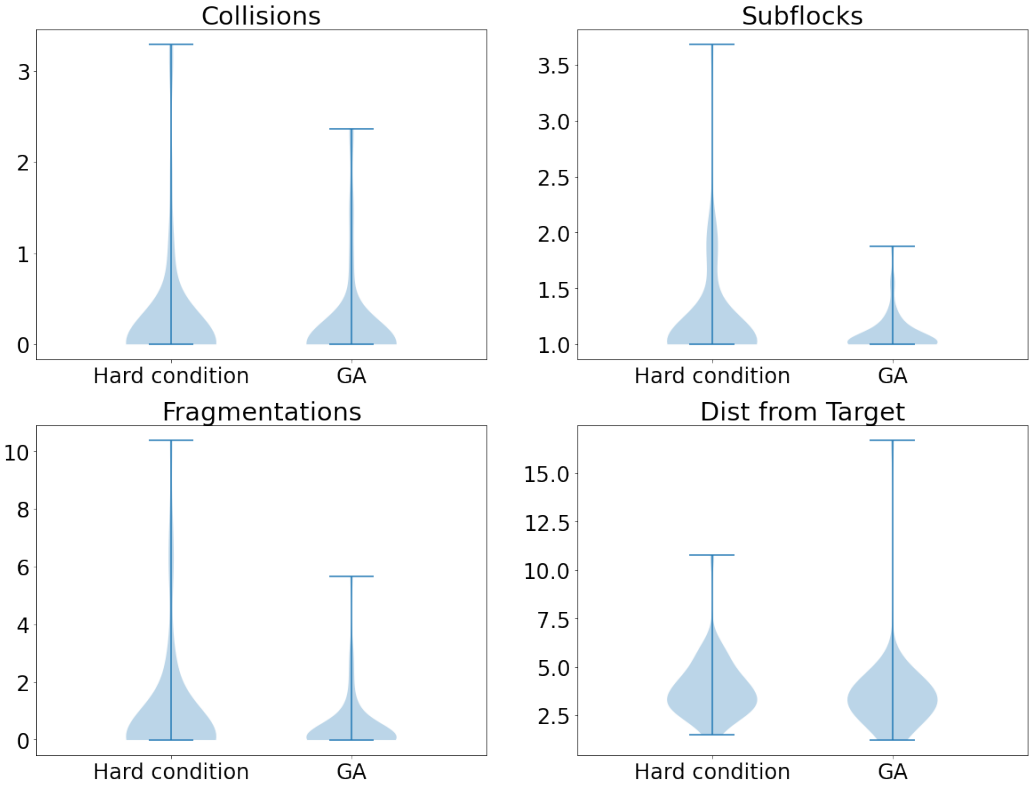


Figure 20: Violin plot comparing metrics of hard condition implementation and GA standout solution 2. To reproduce, run comparisonRuns.ipynb in 'final_version' branch with appropriate list of seeds.

As seen in violin, most of the data points of each of the metrics clump down at the bottom of the plots near zero. This is expected, as most of the randomly seeded paths are easier paths with less obstacles to avoid. This means that both hard condition and GA solutions performed similar as there is not much to compare. Conversely, the top of the violin plots show more of a difference and reflects the more difficult paths that the swarm had to navigate through. Here it is apparent that the GA solutions outperform the original hard condition implementation in collisions, subflocks and fragmentations. However, the distance from virtual leader metric is more contentious, as it seems to fluctuate from environment to environment and solution to solution. This may be because the distance score can easily blow up if the swarm for some reason loses the leader near the beginning of the simulation. Therefore, as the simulation progresses, the distance score will only increase, which causes an outlier in the dataset. This begs the question whether a better metric or simply more metrics should be included to better judge swarm performance.

All in all, it is important not to delve too deep into just one of the metrics. For example, the distance from virtual leader metric only measures the shortest distance from any of the swarm agents to the virtual leader at every time-step. That means that even if the rest of the swarm is completely lost and scattered, as long as one swarm agent is properly following the virtual leader, the swarm would obtain a 'good' score for this metric, even though swarm performance overall is not considered 'good'. This logic can be applied to all of the metrics, as some of the 'best' GA solutions tended to hone a specific metric rather than becoming a 'jack of all trades'. As mentioned previously, this was very apparent in the distance metric which is why later on its weight was decreased even more-so than previously.

Therefore, the four metrics should be judged holistically as a better indicator of swarm performance, in addition to checking how the swarm actually performs in the Mesa simulation visualiser.

4 Suggested Future Work and Conclusion

Due to the long process and various techniques used in this project, there are numerous areas to potentially improve on with future research.

4.1 Feature Selection

Although lots of different manual and feature selection approaches were tried here, there could still be a better combination of features with better analysis. It is still surprising that features representing previous agent states were not as important as speculated, and perhaps there are even better features to collect that were not thought of. At a certain point however, the more complex the feature becomes to collect (such as average change in angle within last 100 steps), the more convoluted it becomes. It is not entirely evidence whether more complex or simple features are required to better suggest agent behaviour, or perhaps combination of both.

As mentioned, increasing the number of features was generally counterproductive, and although an optimal number of features could not be found with consistent results, there may be a better technique in order to do so.

4.2 Clustering

Clustering has several potential improvements, such as trying more clustering algorithms and further fine-tuning the hyperparameters of all clustering algorithms. The algorithms used in this project were only from the sci-kit learn clustering suite, so there are potentially better ones to try from other libraries, however trial and error with hyperparameters may be the priority.

4.3 Genetic Algorithm

The big avenue for further research using the GA to map clusters to modes would be to try the BIRCH clustering solutions which showed promise. The centroids data has already been extracted and saved to brc_centroids.csv.

Since the GA is pretty straight-forward in its use, optimising its hyperparameters or changes its methods (cross, mutation, selection, culling) could speed up the search for the global maximum over the solution space. This can lead to overall better solutions but is still affected by chance.

The fitness equation and calculation could also see improvements. The weights can always be adjusted along with changing and adding any metrics that one sees fit to better encapsulate the fitness of a swarm's performance. However, instead of a weighted sum, the fitness calculation could simply take the maximum fitness component and set that as its overall fitness. This is a more 'pessimistic' approach and highlights each solution's worst component. For example, no matter how low collisions, subflocks and fragmentations are, if distance from leader is extremely high, the overall fitness is not desired. This method would prefer solutions which have very low values in all components, such that when the minimum is taken, it's overall fitness is still low, resulting in a good solution with good metrics.

4.4 Comparing with Hard Condition Behaviour Selection

The final comparison runs when comparing standout solutions to hard condition implementation could be visually improved. Hard condition runs can be plotted against multiple standout solutions so that even the solutions themselves can be compared on the same seeds. Currently they the solutions are only compared to the GA solutions in their selected seeds. Additionally, the comparison metrics can be separated into simple and difficult seeds/paths to visually verify that the standout solutions do indeed outperform in harder paths.

Comparisons in run-time should also be explored. This new behaviour switching approach requires gathering the selected agent features and calculating the nearest centroid based on features at every time step, whereas the original method only needs to check whether a maximum of 3 thresholds have been passed before switching. This may greatly increase compute especially when increasing swarm size as these calculations have to be performed for each swarm agent individually. Therefore, this should be investigated in simulation before even considering real-time applications.

4.5 Conclusion

There is reason to believe that an Autonomously Learned Behaviour Selection system can outperform hard conditions to dictate behaviour switching in Multi-Modal Flocking Swarms. Although metrics and visual simulations have shown that some many-to-one mappings of feature clusters to agent modes outperform the original implementation in more complex environments, these results must be verified at all steps.

5 Appendix

5.1 Appendix 1: Hard Condition Behaviour Selection Against Standout Solution 1 and Solution 2 in 50 randomly seeded environments

Seed	Colls	Soln Colls	Subfs	Soln Subfs	Frags	Soln Frags	Dist	Soln Dist
2	0.0	0.839	1.003	1.004	0.016	0.032	5.259	3.867
4	0.809	2.363	1.008	1.003	0.082	0.023	2.428	2.812
5	0.0	0.948	1.001	1.012	-0.0	0.032	4.255	3.795
6	0.0	0.0	1.001	1.001	-0.0	-0.0	3.048	2.582
7	0.26	0.122	1.405	1.08	3.312	0.773	4.528	3.557
8	0.0	0.0	1.242	1.14	0.231	0.093	4.528	3.98
9	0.0	0.0	1.074	1.099	0.117	0.109	2.751	3.581
12	0.0	0.0	1.07	1.011	0.403	0.059	3.106	3.172
14	0.0	0.0	1.002	1.003	0.015	0.03	3.101	2.417
15	0.0	0.0	1.003	1.056	0.027	0.622	3.917	4.23
18	0.0	0.0	1.002	1.055	0.015	0.049	5.554	3.555
19	0.0	0.0	1.002	1.012	0.015	0.116	4.264	3.822
24	0.0	0.0	1.002	1.002	0.015	0.015	2.27	2.57
26	0.0	0.0	1.001	1.003	-0.0	0.008	1.973	1.928
27	0.0	0.0	1.001	1.011	-0.0	0.051	3.978	3.1
28	0.137	0.0	1.844	1.128	2.389	0.653	4.492	3.972
31	0.0	0.0	1.002	1.002	0.015	0.015	3.257	2.259
32	0.0	0.0	1.001	1.001	-0.0	-0.0	3.908	3.661
33	0.72	0.296	2.133	1.192	2.818	0.585	4.299	3.998
34	0.0	0.0	1.001	1.004	-0.0	0.034	2.789	2.763
35	0.0	0.0	1.007	1.004	0.076	0.034	2.766	3.214
38	0.0	0.0	1.002	1.002	0.015	0.015	2.902	1.219
39	0.0	0.0	1.0	1.0	-0.015	-0.015	2.912	2.183
40	0.0	0.0	1.001	1.003	-0.0	0.007	2.142	2.373
41	3.175	0.803	2.036	1.226	7.578	2.685	3.453	4.129
42	0.0	0.0	1.267	1.677	2.021	3.062	3.544	3.719
47	0.0	0.0	1.003	1.004	0.03	0.042	3.372	3.011
50	0.0	0.0	1.01	1.006	0.134	0.033	3.113	2.199
52	0.0	0.0	1.0	1.005	-0.015	0.051	2.834	3.218
53	0.686	0.0	2.148	1.057	5.846	0.62	4.9	4.077
54	0.0	0.0	1.002	1.004	0.015	0.016	2.695	2.969
56	0.023	0.035	1.108	1.105	0.042	0.207	5.614	4.43
60	0.0	0.0	1.088	1.186	0.433	0.843	6.031	5.821
61	0.0	0.0	1.001	1.003	-0.0	0.015	2.212	1.725
64	0.0	0.0	1.002	1.357	0.006	0.912	3.756	3.38
66	0.0	0.0	1.0	1.0	-0.015	-0.015	3.208	3.001
67	0.0	0.445	1.001	1.001	-0.0	-0.0	3.1	3.355
68	0.656	2.29	1.24	1.585	1.947	2.591	2.687	3.17
71	0.0	0.0	1.016	1.235	0.105	0.306	5.919	4.551
73	0.0	0.0	1.041	1.088	0.021	0.152	6.633	5.304
75	0.18	0.049	1.429	1.384	1.087	0.833	10.297	20.813
78	1.139	2.118	2.204	1.263	6.057	2.356	5.466	4.339
80	0.0	0.0	1.001	1.003	-0.0	0.021	3.102	3.113
81	0.071	0.0	1.033	1.01	0.058	0.047	3.615	4.411
83	0.0	0.0	1.003	1.003	0.03	0.03	1.512	1.219
84	0.0	0.0	1.0	1.0	-0.015	-0.015	3.666	2.887
85	0.993	0.0	1.018	1.003	0.189	0.019	4.235	4.125
87	0.0	0.0	1.0	1.0	-0.015	-0.015	2.742	1.58
88	0.0	0.0	1.001	1.055	-0.0	0.011	4.369	4.834
89	4.799	2.897	2.967	1.82	8.713	5.251	51.177	4.584