Evolving Deep Recurrent Neural Networks Using Ant Colony Optimization

Travis Desell, Sophine Clachar

Department of Computer Science
University of North Dakota

James Higgins, Brandon Wild

Department of Aviation
University of North Dakota

The 15th European Conference on Evolutionary Computation in Combinatorial Optimization

April 08-10, 2015 Copenhagen, Denmark

Overview

Introduction:
Ant Colony Optimization

Introduction: Ant Colony Optimization

Ant colony optimization is based off the behavior of swarming ants. Ants deposit pheromone when they have found a good path, and other ants are drawn to those pheromones. Over time the pheromone degrades on paths not frequently traveled. This makes it a very interesting algorithm for determining paths through graphs.

"Safari ants" by Mehmet Karatay - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Safari_ants.jpg#mediaviewer/File:Safari_ants.jpg

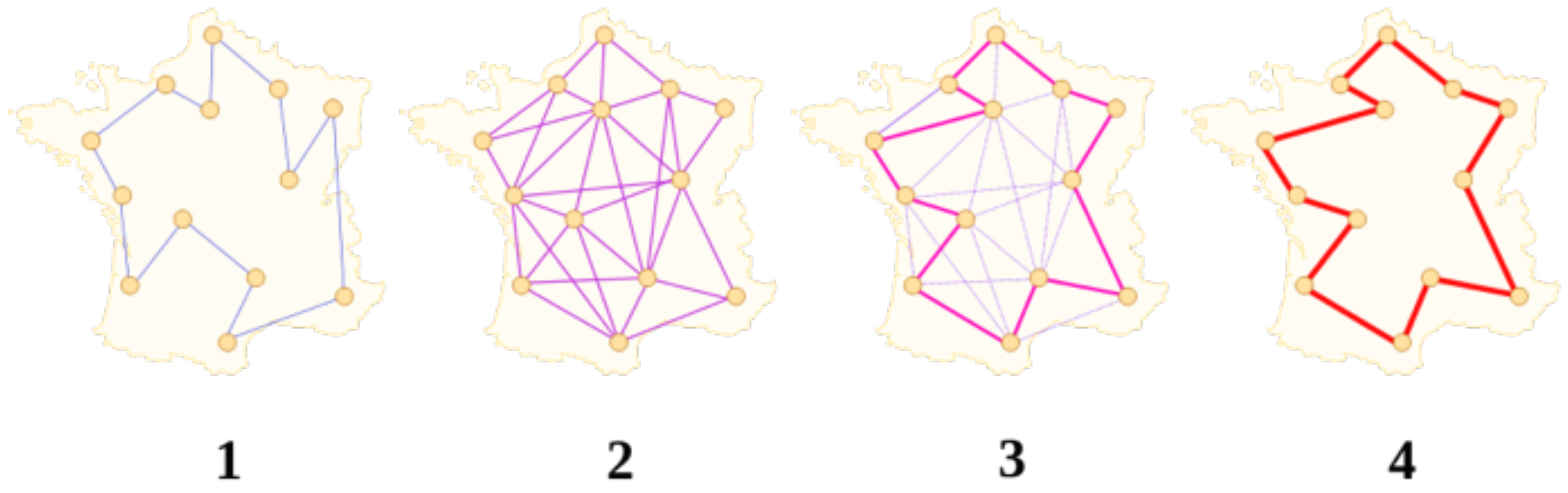


Image is: "Aco TSP". Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Aco_TSP.svg#mediaviewer/File:Aco_TSP.svg

Introduction: Ant Colony Optimization

While ant colony optimization (ACO) has seen wide use on problems such as the Traveling Salesman [1], and has even been extended to training the weights of neural networks [2-5], it has not yet been used to evolve the structure of neural networks.

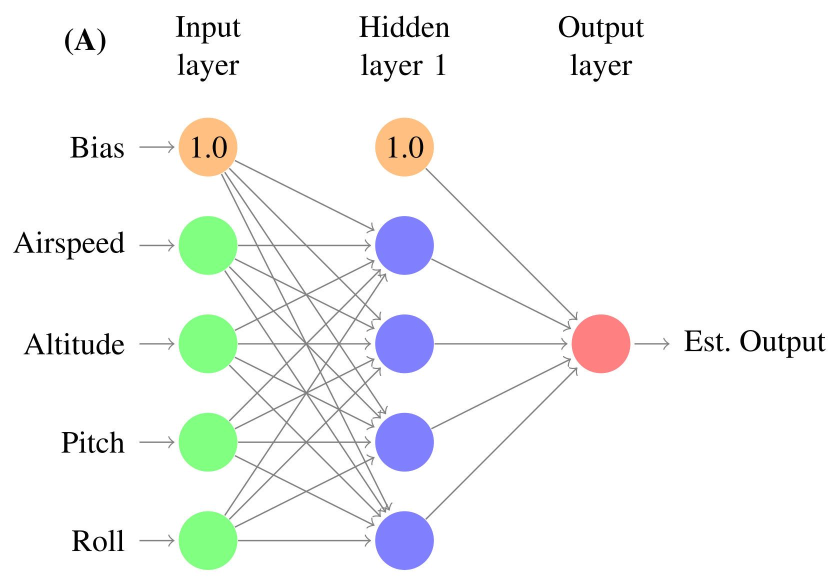
This is particularly interesting, as designing neural networks is in many ways a graph path selection problem, which is what ACO was originally designed for.

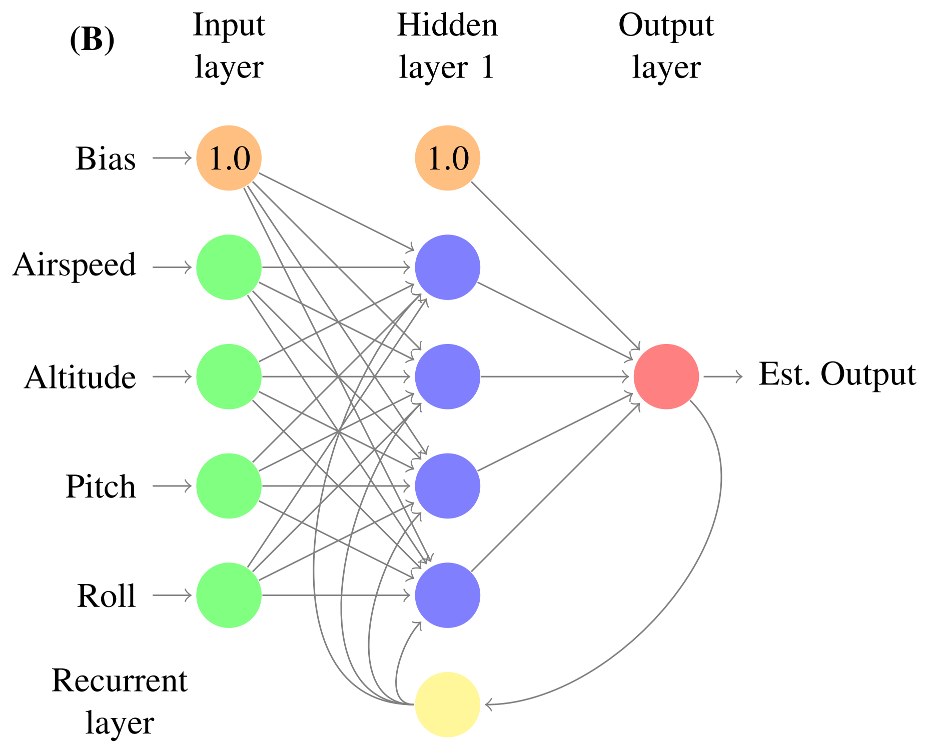
1. M.Dorigo and L. M. Gambardella. Ant colonies for the traveling salesman problem. Bio Systems, 43(2):73–81, 1997.
2. J.-B. Li and Y.-K. Chung. A novel back-propagation neural network training algorithm designed by an ant colony optimization. In Transmission and Distribution Conference and Exhibition: Asia and Pacific, 2005 IEEE/PES, pages 1–5. IEEE, 2005.
3. C.Blum and K. Socha. Training feed-forward neural networks with ant colony optimization: An application to pattern classification. In Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on, pages 6–pp. IEEE, 2005.
4. A. Pandian. Training neural networks with ant colony optimization. PhD thesis, California State University, Sacramento, 2013.
5. M. Unal, M. Onat, and A. Bal. Cellular neural network training by ant colony optimization algorithm. In Signal Processing and Communications Applications Conference (SIU), 2010 IEEE 18th, pages 471–474. IEEE, 2010.

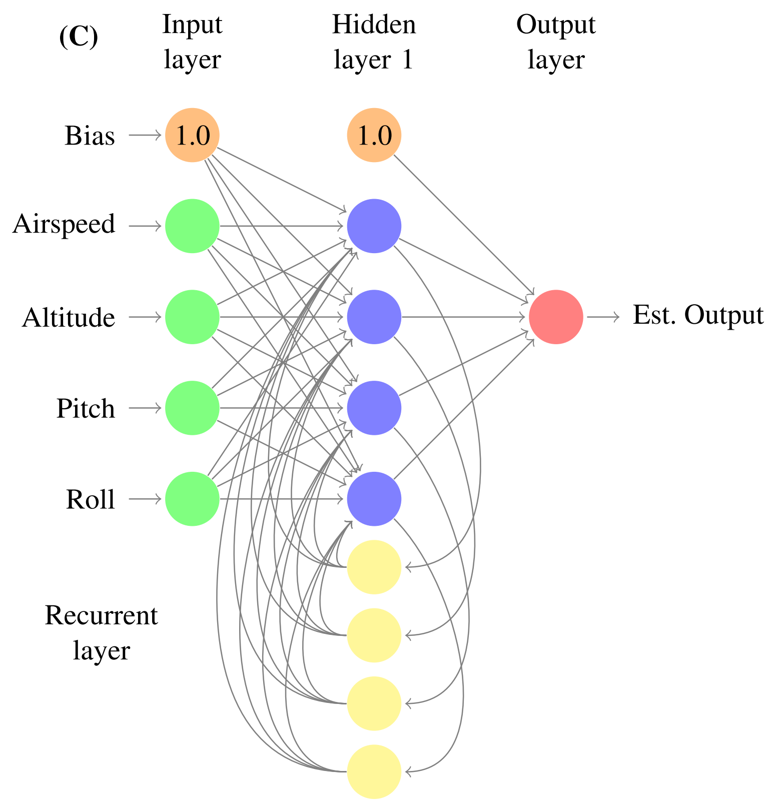
Introduction:
Recurrent Neural Networks

Introduction: Recurrent Neural Networks

Recurrent neural networks (B and C) are a variation on neural networks which employ "memory" neurons and/or backward links, unlike feed forward neural networks which only have forward links (A).







They have been widely used for time series data prediction [6,7].

6. S.F.Crone, M.Hibon,and K.Nikolopoulos. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting*, 27(3):635–660, 2011.

7. G. P. Zhang. Neural networks for time-series forecasting. In *Handbook of Natural Computing*, pages 461–477. Springer, 2012.

Introduction: Recurrent Neural Networks

Unfortunately, RNNs for time series data prediction typically require few input nodes (<20), so convolutional neural networks are not applicable.

Further, the recurrent links and memory neurons provide significant training challenges, so pre-training strategies based on Restricted Boltzmann Machines [8] and auto-encoders [9] are also not applicable.

8. Hinton, Geoffrey, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation* 18.7 (2006): 1527-1554.
9. Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1 (2009): 1-127.

Introduction: Recurrent Neural Networks

Further than this, it is even challenging to get backpropagation working to train a recurrent neural network. In order to do backpropagation, the neural network needs to be "unrolled" for every time step, leading to a massive neural network with shared weights:

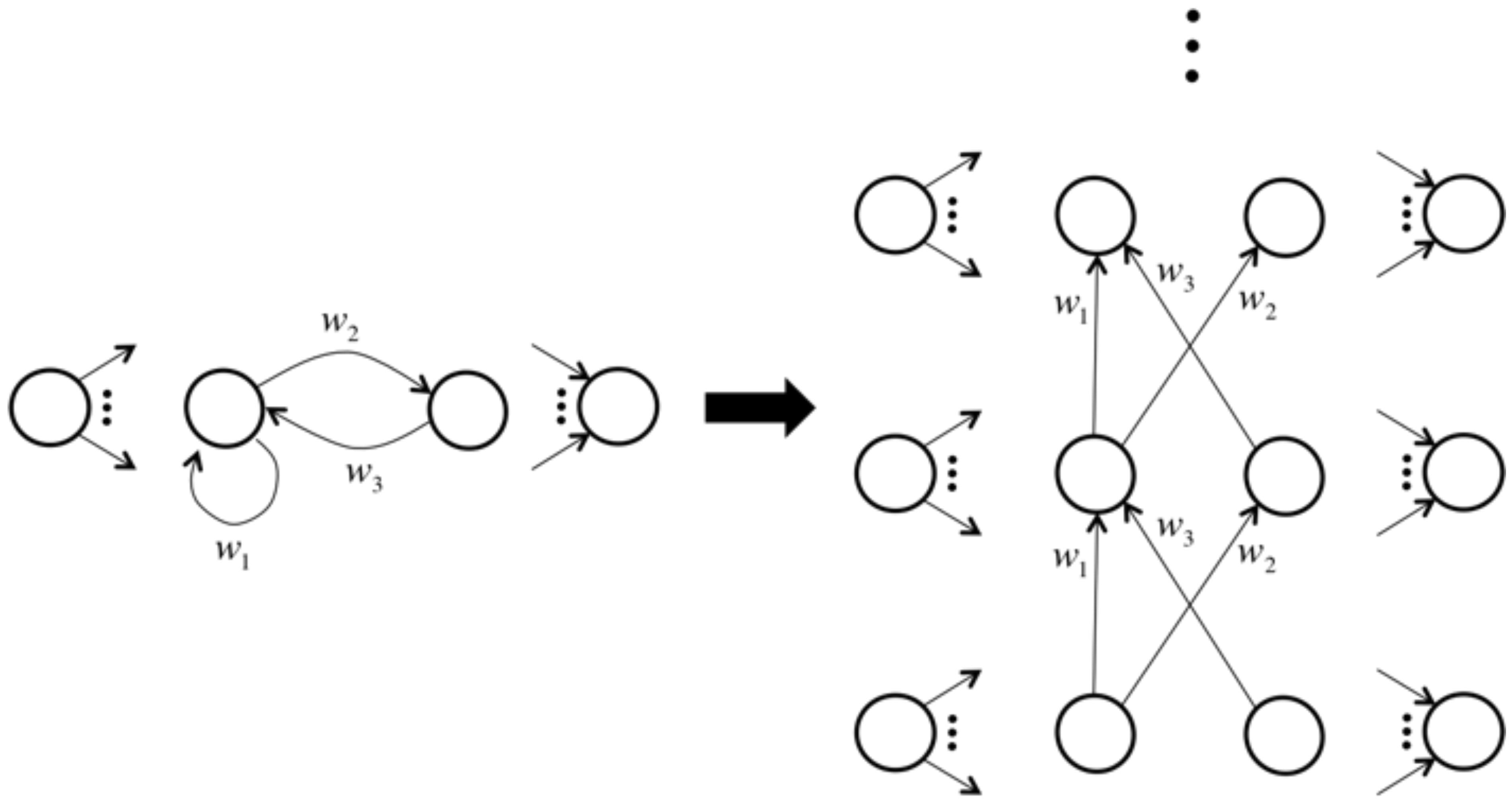


Image from: http://nikhilbuduma.com/2015/01/11/a-deep-dive-into-recurrent-neural-networks/

Introduction: Recurrent Neural Networks

These large neural networks become very challenging to train with backpropagation, leading to problems of *exploding* and *vanishing gradients*. There has been recent work in developing techniques to overcome this using clipped gradients [10], long short term memory (LSTM) neurons [11] and other gradient-free methods [12].

As such, evolutionary optimization strategies for continuous parameters such as Particle Swarm Optimization (PSO) and Differential Evolution (DE) also show much promise in training RNNs, as they do not require unrolling or the calculation of gradients.

10. Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "Understanding the exploding gradient problem," Tech. Rep., Universite´ De Montre´al, 2012, arXiv:arXiv:1211.5063.
11. Graves, Alex, et al. "A novel connectionist system for unconstrained handwriting recognition." Pattern Analysis and Machine Intelligence, IEEE Transactions on 31.5 (2009): 855-868.
12. Bengio, Yoshua, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. "Advances in optimizing recurrent networks." Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.

Introduction:
Neuro-Evolution

Introduction: Neuro-Evolution

While there are some standard architectures for recurrent neural networks (such as the previously shown Jordan and Elman RNNs), it is still an open question as to what design of an RNN will provide the best predictions.

Neuro-Evolution [13-14] is the process of *evolving* the structure of neural networks as opposed to using fixed neural network architectures. There have been successful strategies for this, such as NeuroEvolution of Augmenting Topologies (NEAT) [15] and HyperNEAT [16], however while these can evolve backward links, they cannot evolve "memory" neurons as used in time series data prediction without some modification.

13. Annunziato, M., M. Lucchetti, and S. Pizzuti. "Adaptive Systems and Evolutionary Neural Networks: a Survey." Proc. EUNITE02, Albufeira, Portugal (2002).
14. Floreano, Dario, Peter Dürr, and Claudio Mattiussi. "Neuroevolution: from architectures to learning." Evolutionary Intelligence 1.1 (2008): 47-62.
15. K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2):99–127, 2002.
16. K. O. Stanley, D. B. D’Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. Artificial life, 15(2):185–212, 2009.

Motivation:
Flight Data Prediction

Motivation: Flight Data Prediction

General aviation comprises 63% of all civil aviation activity in the United States; covering operation of all non-scheduled and non-military aircraft [17,18].

While general aviation is a valuable and lucrative industry, it has the highest accident rates within civil aviation [19].

For many years, the general aviation accident and fatality rates have hovered around 7 and 1.3 per 100,000 flight hours, respectively [20].

17. B. Elias. Securing general aviation. DIANE Publishing, 2009.
18. K. I. Shetty. Current and historical trends in general aviation in the United States. PhD thesis, Massachusetts Institute of Technology Cambridge, MA 02139 USA, 2012.
19. National Transportation Safety Board (NTSB), 2012. https://www.ntsb.gov/safety/mwl5_2012.html
20. Aircraft Owners and Pilots Association(AOPA), January 2014. http://www.aopa.org/About-AOPA/Statistical-Reference-Guide/General-Aviation-Safety-Record-Current-and-Historic.aspx

Motivation: Flight Data Prediction

State of the art in aviation is in many ways primitive when it comes to accident prevention. In many cases it is *reactionary* -- after an accident occurs an investigation is performed and depending on the results of the investigation, new regulations are proposed to prevent further incidents of a similar nature.

Likewise, current state of the art in flight data analysis deals with *exceedences*, i.e., "did altitude exceed a threshold?" If this happens, then the flight will potentially be reviewed.

Motivation: Flight Data Prediction

Having the ability to *predict* flight parameters based on multiple other parameters as input is a first step towards developing sensors which can intelligently detect anomalous behavior or predict accident precursor behavior. Bringing machine learning strategies into flight data analysis and accident prediction has great potential for preventing future accidents in a proactive manner.

Further, these same strategies can be used to predict and prevent hardware failures or suggest pre-emptive maintenance, reducing costs for airlines.

Motivation:
The National General Aviation Flight Database

Motivation: The National General Aviation Flight Database

The National General Aviation Flight Information Database (NGAFID) has been developed at the University of North Dakota as a central repository for general aviation flight data. It consists of per-second flight data recorder (FDR) data from three fleets of aircraft.

As of November 2014, the database stores FDR readings from over 200,000 flights, with more being added daily. It currently stores over 750 million per-second records of flight data. The NGAFID provides an invaluable source of information about general aviation flights, as most of these flights are from aviation students, where there is a wider variance in flight parameters than what may normally be expected within data from professionally piloted flights.

Motivation: The National General Aviation Flight Database

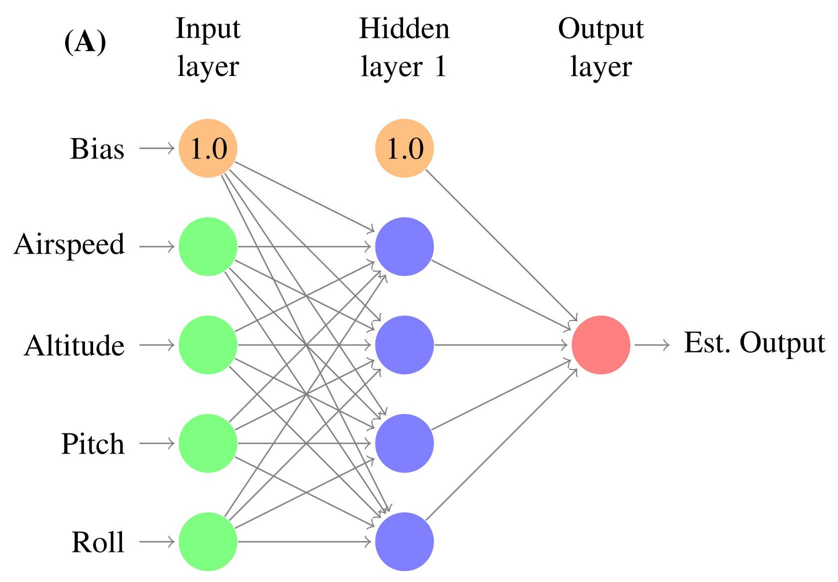
Time series flight data for this work were obtained from the NGAFID, and they have been made publicly available as a data release:

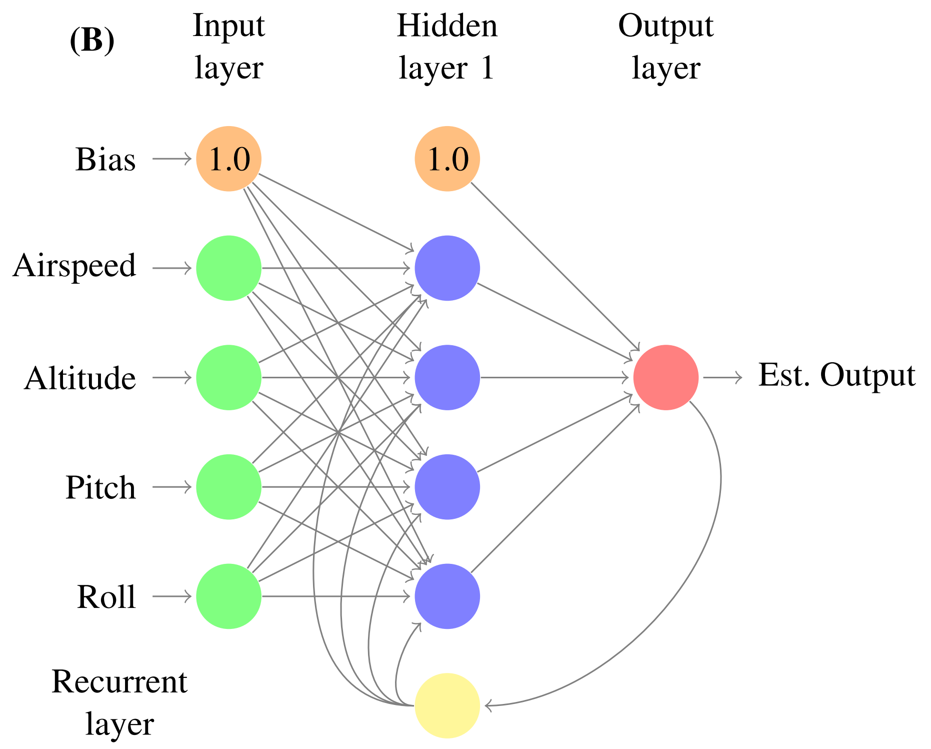
http://people.cs.und.edu/~tdesell/ngafid_releases.php

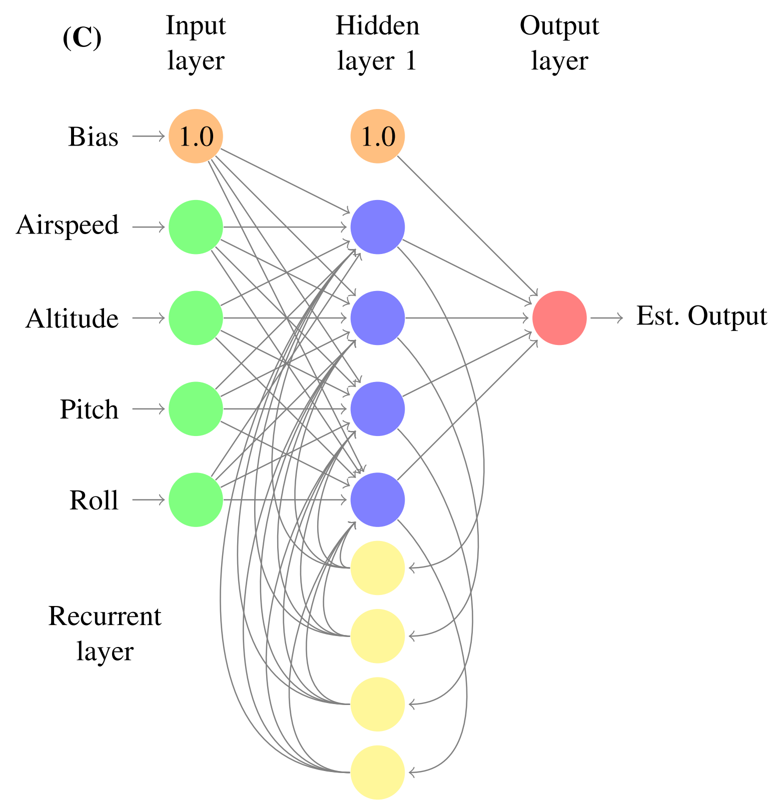
Previous Results

Previous Results

We trained 15 different recurrent neural network designs (variations on the feed forward, Jordan and Elman RNNs below) for four different output parameters for five different flights. Training for all 4 output parameters concurrently was shown to not be feasible, and asynchronous differential evolution outperformed backpropagation. In total, 12,000 high performance computing runs of differential evolution with various parameters were performed.





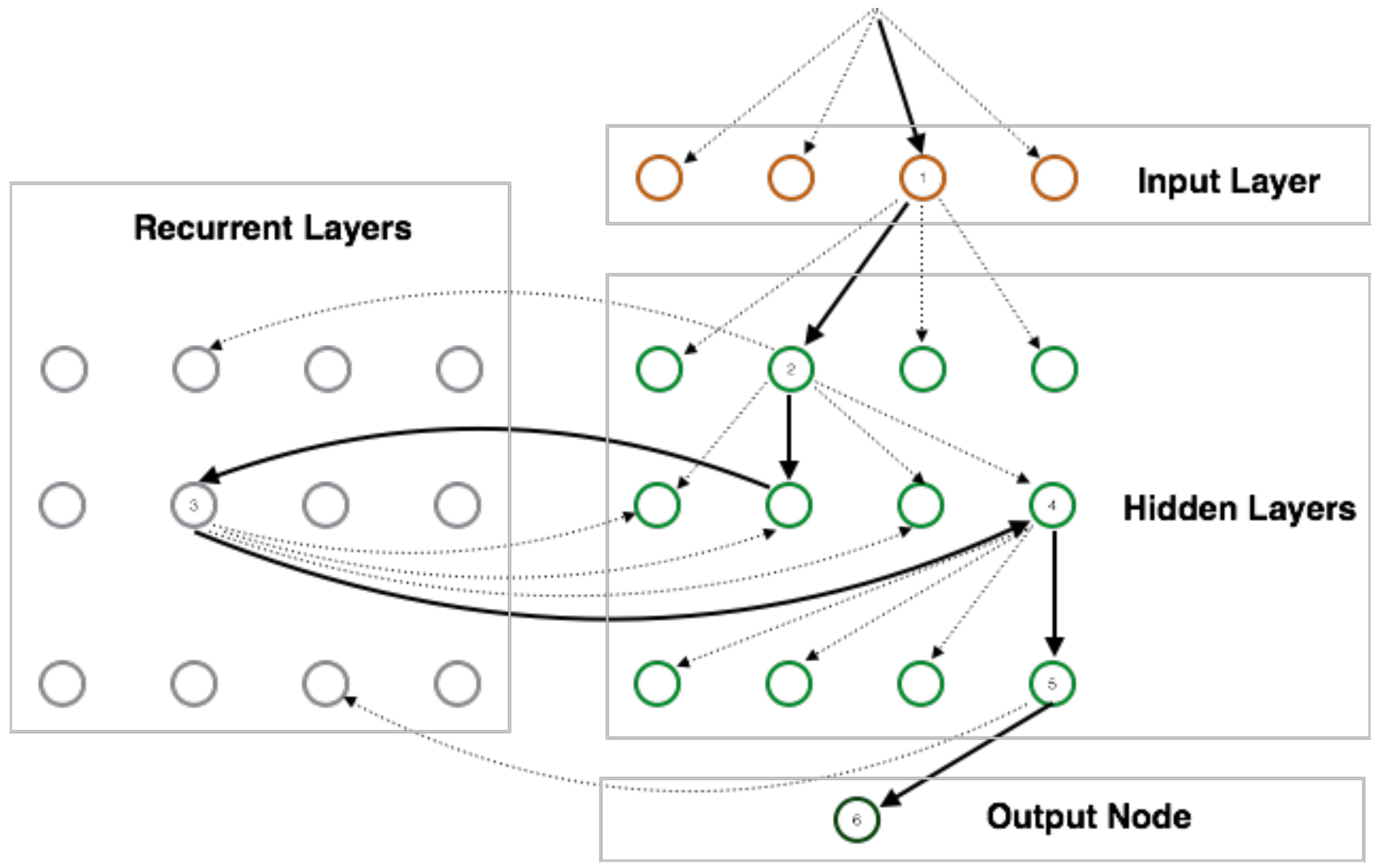


21. T. Desell, S. Clachar, J. Higgins, and B. Wild. Evolving neural network weights for time-series prediction of general aviation flight data. In T. Bartz-Beielstein, J. Branke, B. Filipi, and J. Smith, editors, Parallel Problem Solving from Nature PPSN XIII, volume 8672 of Lecture Notes in Computer Science, pages 771–781. Springer International Publishing, 2014.

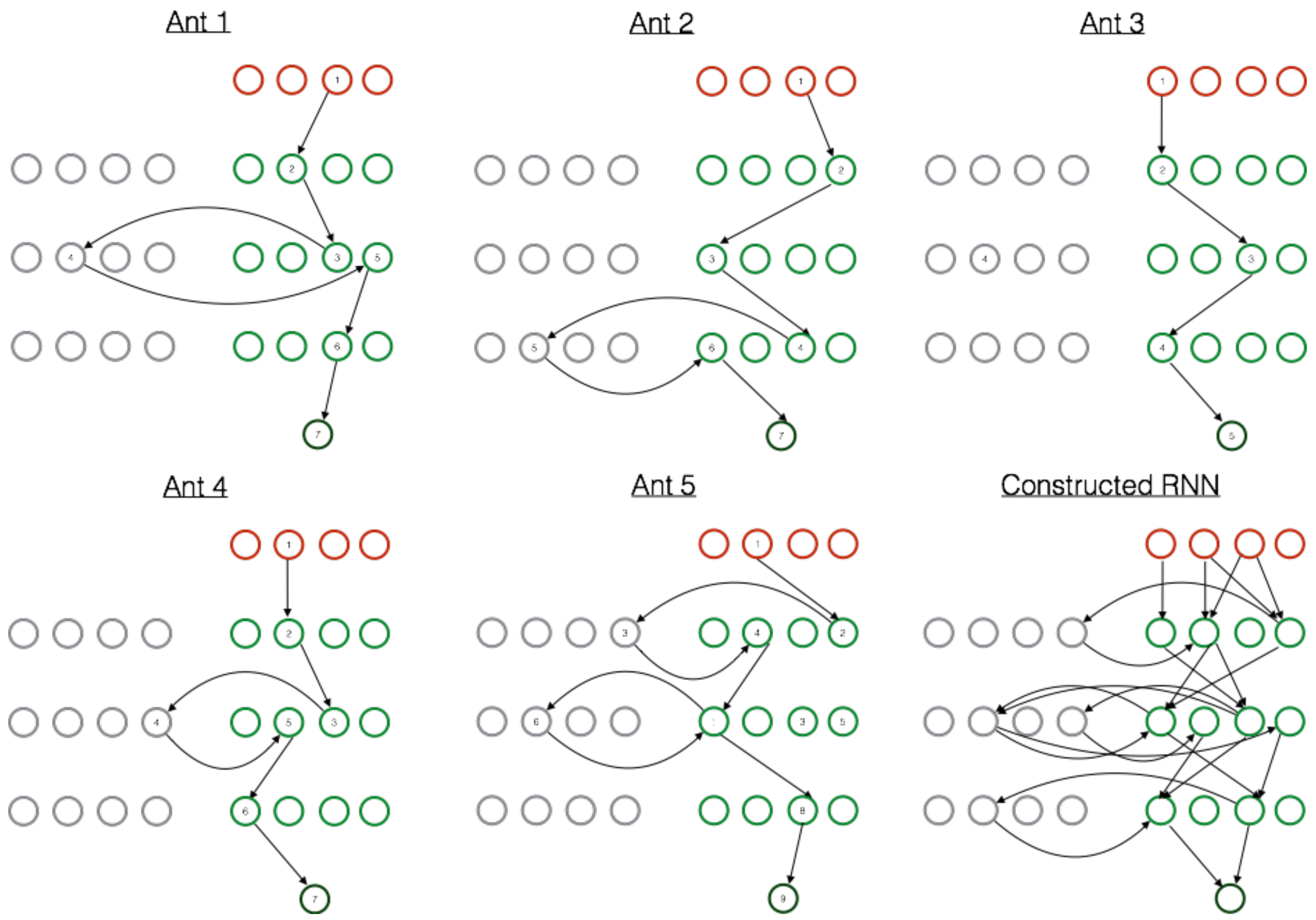
Methodology

Methodology

Ants select a random path through the possible RNN nodes based on the number of pheromones on each link. Ants are allowed to only visit one recurrent node per layer.



Methodology



The multiple ant paths are combined into a single recurrent neural network.

Methodology

RNNs can be trained by any training method (backprop, particle swarm, differential evolution, etc.) A population of the best found RNNs is kept. When a new best RNN is found, pheromones are incremented along all edges of that RNN.

After each evaluation of an RNN, all weights are degraded by a small pheromone degradation weight (1%, 5% and 10% were tested).

RNNs can be evaluated asynchronously on distributed compute nodes, with a single master process handling the generation of new RNNs, pheromone increments/decrements, and the population of best found RNNs.

Results

Results: Optimization Software

The numerical optimization strategies and ant colony optimization algorithm have been implemented as part of the Toolkit for Asynchronous Optimization (TAO) which is freely available on github:

https://github.com/travisdesell/tao

TAO is implemented in C++ and MPI for efficiency and easy use on clusters and supercomputers, as well as for volunteer computing systems such as BOINC.

Results: Experiments

As prior results [21] have shown that backpropagation does not perform well, this work focused on using particle swarm optimization (PSO) to train the evolved neural networks. Tests with differential evolution found both strategies to perform similarly.

PSO was used with a population of 200, an inertia weight of 0.75, and local and global best constants of 1.5 for all runs. It was run for 250, 500 and 1000 iterations for the different ACO runs.

21. T. Desell, S. Clachar, J. Higgins, and B. Wild. Evolving neural network weights for time-series prediction of general aviation flight data. In T. Bartz-Beielstein, J. Branke, B. Filipi, and J. Smith, editors, Parallel Problem Solving from Nature PPSN XIII, volume 8672 of Lecture Notes in Computer Science, pages 771–781. Springer International Publishing, 2014.

Results: Experiments

The ACO strategy was used to train neural networks with 3, 4 and 5 hidden layers (with a recurrent node for each hidden node), using 4 and 8 hidden nodes per layer.

Pheromone degradation rates were 10%, 5% and 1%, with a minimum pheromone value of 1, and a maximum pheromone value of 20.

The number of ants used was twice the number of hidden nodes per layer (so 8 and 16).

The population size for the ACO strategy was 20.

Results: Experiments

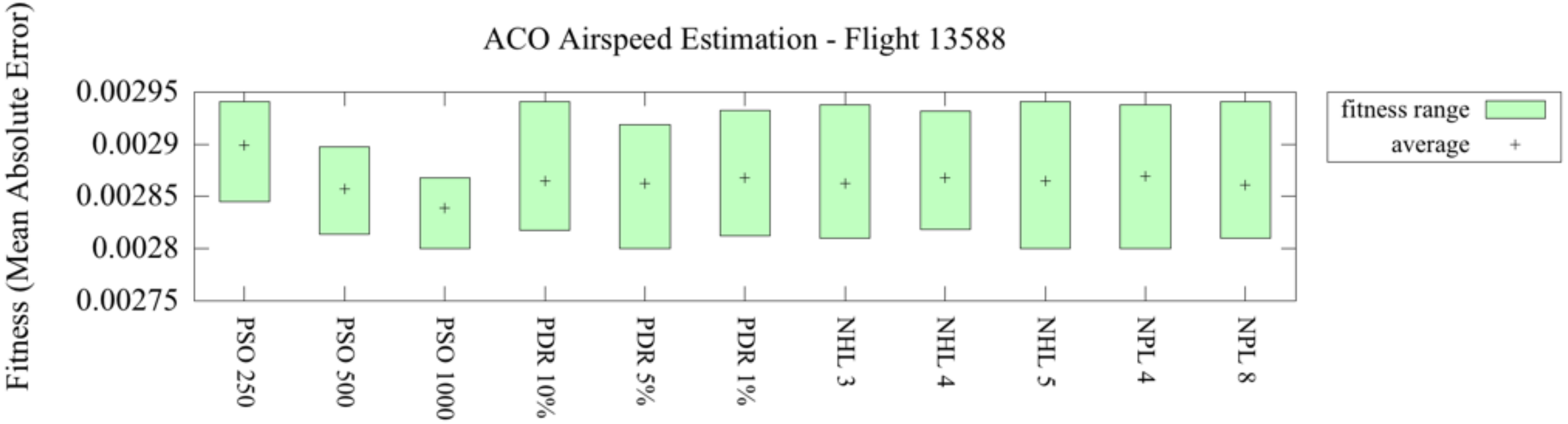
Each run was done allocating 64 processors across 8 compute nodes, and was allowed to train for 1000 evaluations of generated recurrent neural networks.

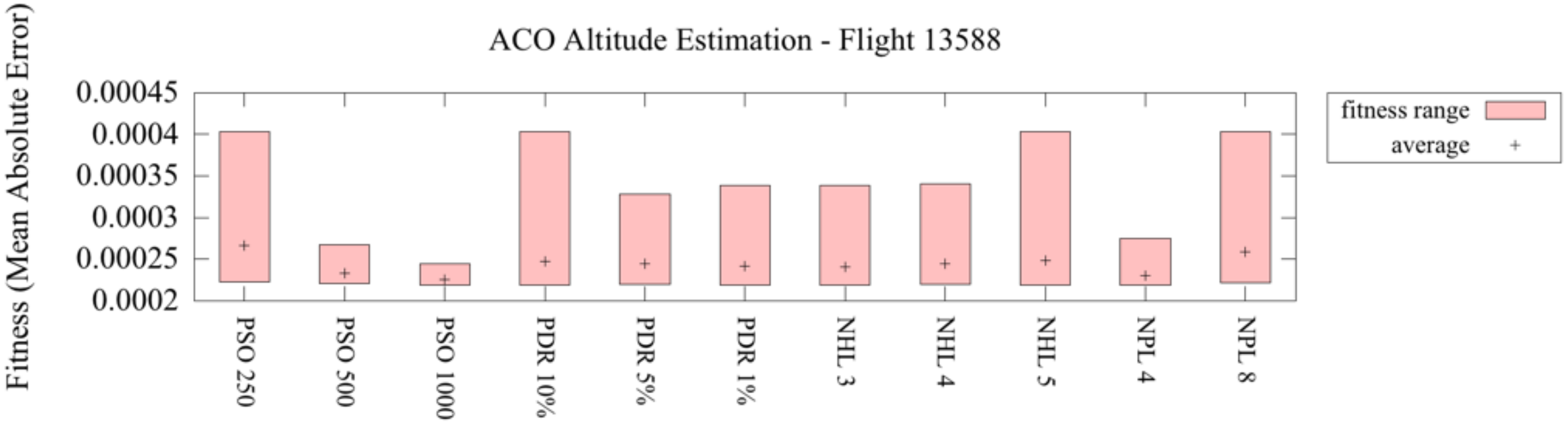
Runs were done with PSO performing 250 iterations (50,000 objective functions) took ~30 minutes, 500 iterations (100,000 objective functions) took ~1 hour, and 1000 iterations (200,000 objective functions) took ~2 hours. Previous results allowed up to 15,000,000 objective function evaluations.

All runs were done on flight ID 13588 from the public data release. All four input parameters (altitude, airspeed, pitch and roll) were used to predict one of those four output parameters. Runs were done generating neural networks for each output parameter separately, as predicting all four with the same network performed poorly in previous work.

Results: ACO Parameter Setting Analysis

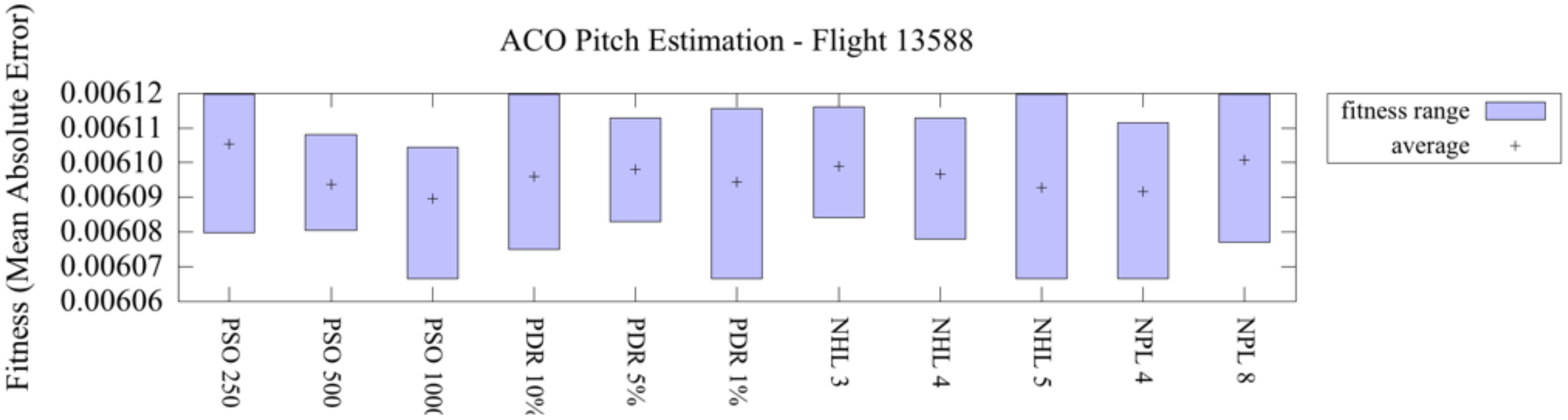
The following figures show the ranges of the best fitnesses found by the ACO evolution given the different meta-parameters used.

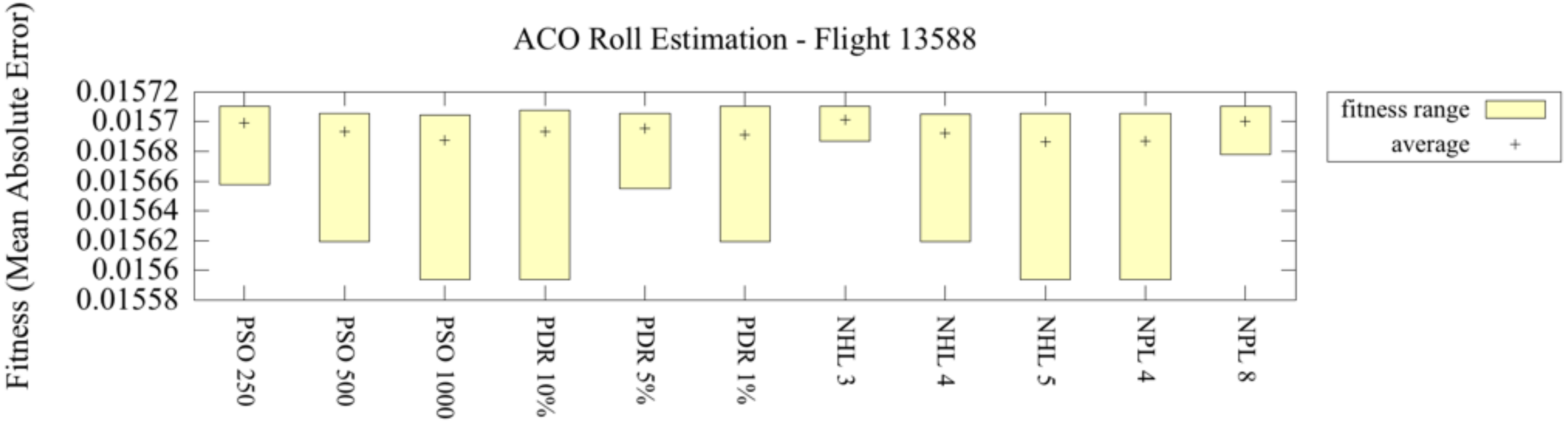




Results: ACO Parameter Setting Analysis

The following figures show the ranges of the best fitnesses found by the ACO evolution given the different meta-parameters used.





Results: ACO Parameter Setting Analysis

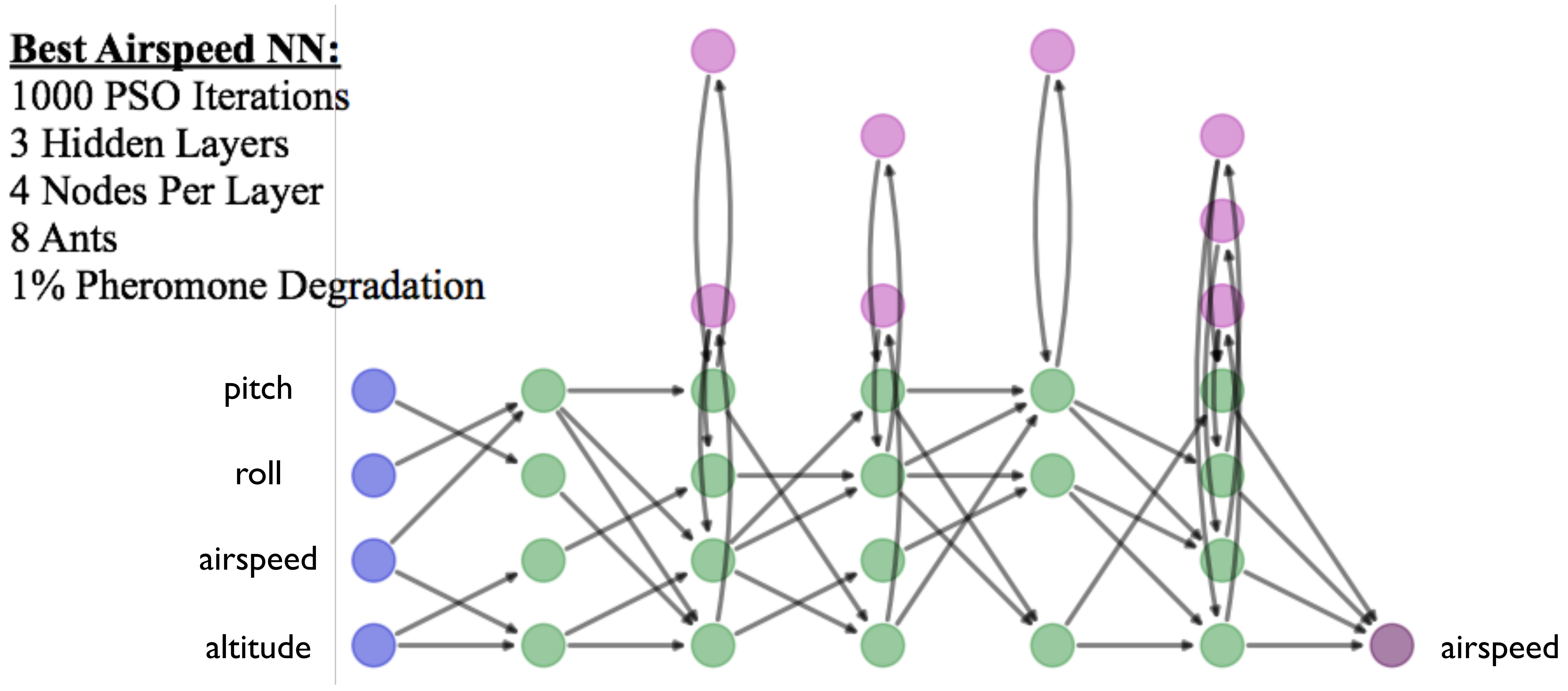
In general, there was a strong correlation between increased PSO iterations and the best fitnesses found.

Across all runs, 4 nodes per layer performed the best, and apart from altitude, 5 hidden layers performed the best.

There did not appear to be a strong trend for the pheromone degradation rate.

Results: Best Found Neural Networks

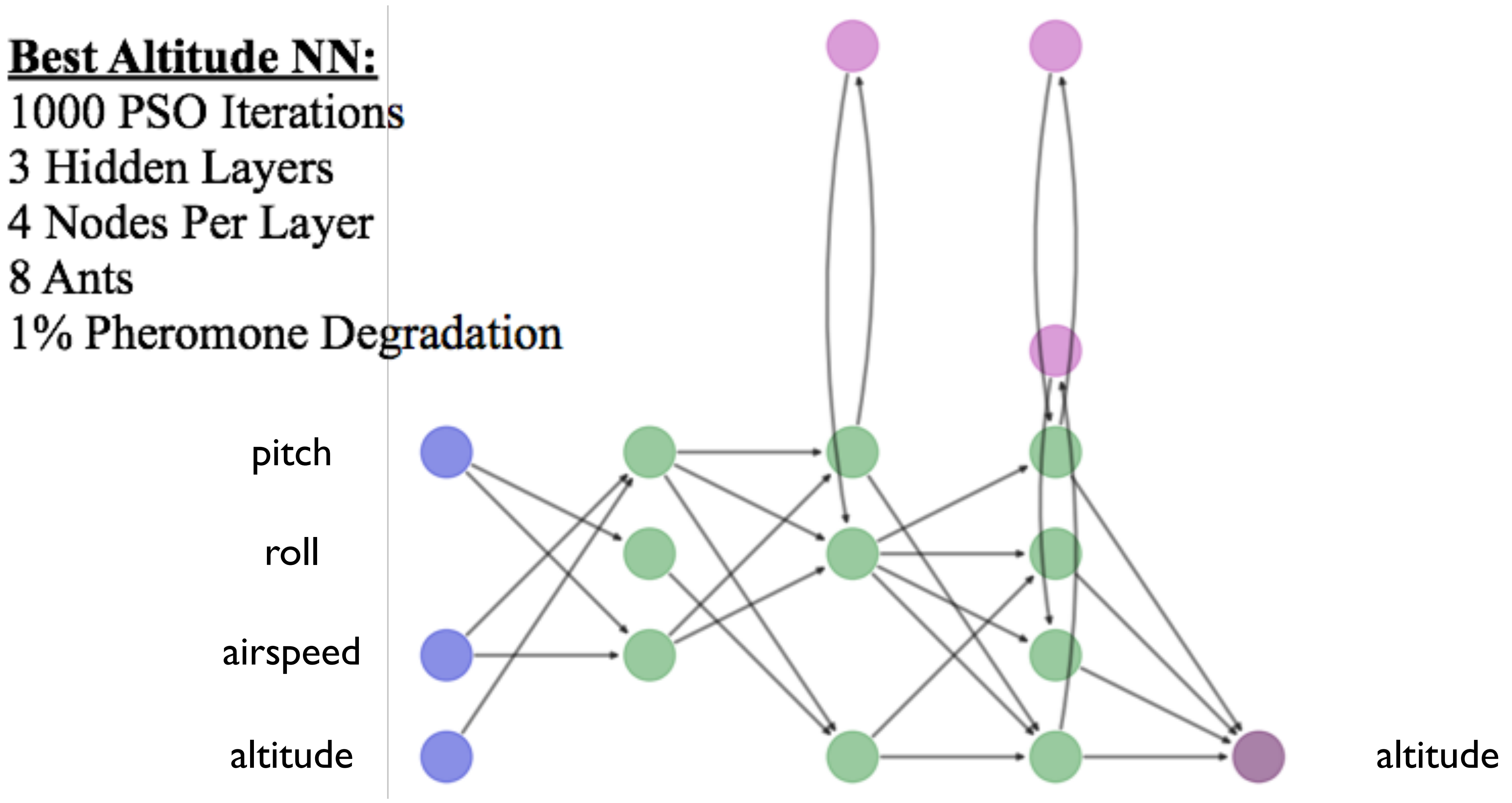
The following shows the best neural network for predicting airspeed as evolved by the ACO strategy.



Best Airspeed NN:
1000 PSO Iterations
3 Hidden Layers
4 Nodes Per Layer
8 Ants
1% Pheromone Degradation

Results: Best Found Neural Networks

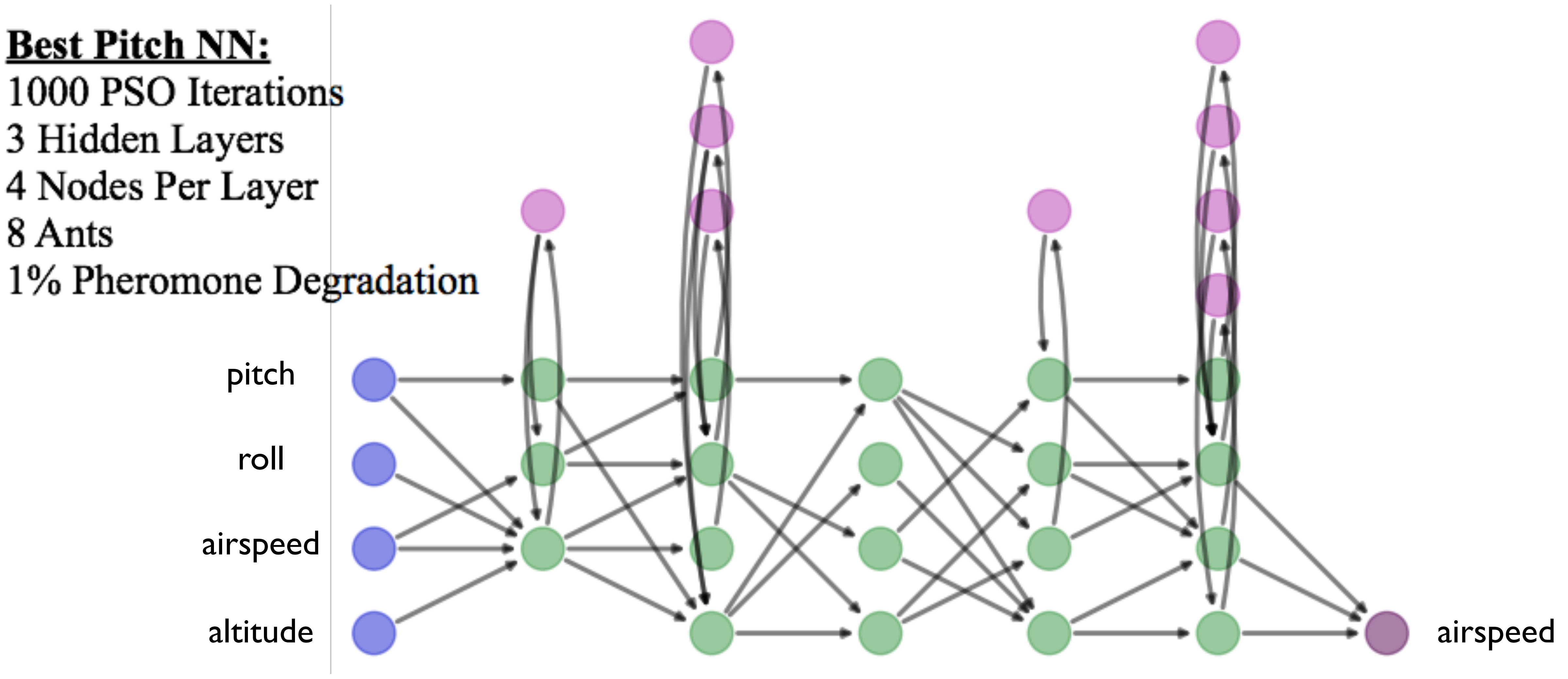
The following shows the best neural network for predicting altitude as evolved by the ACO strategy.



Best Altitude NN:
1000 PSO Iterations
3 Hidden Layers
4 Nodes Per Layer
8 Ants
1% Pheromone Degradation

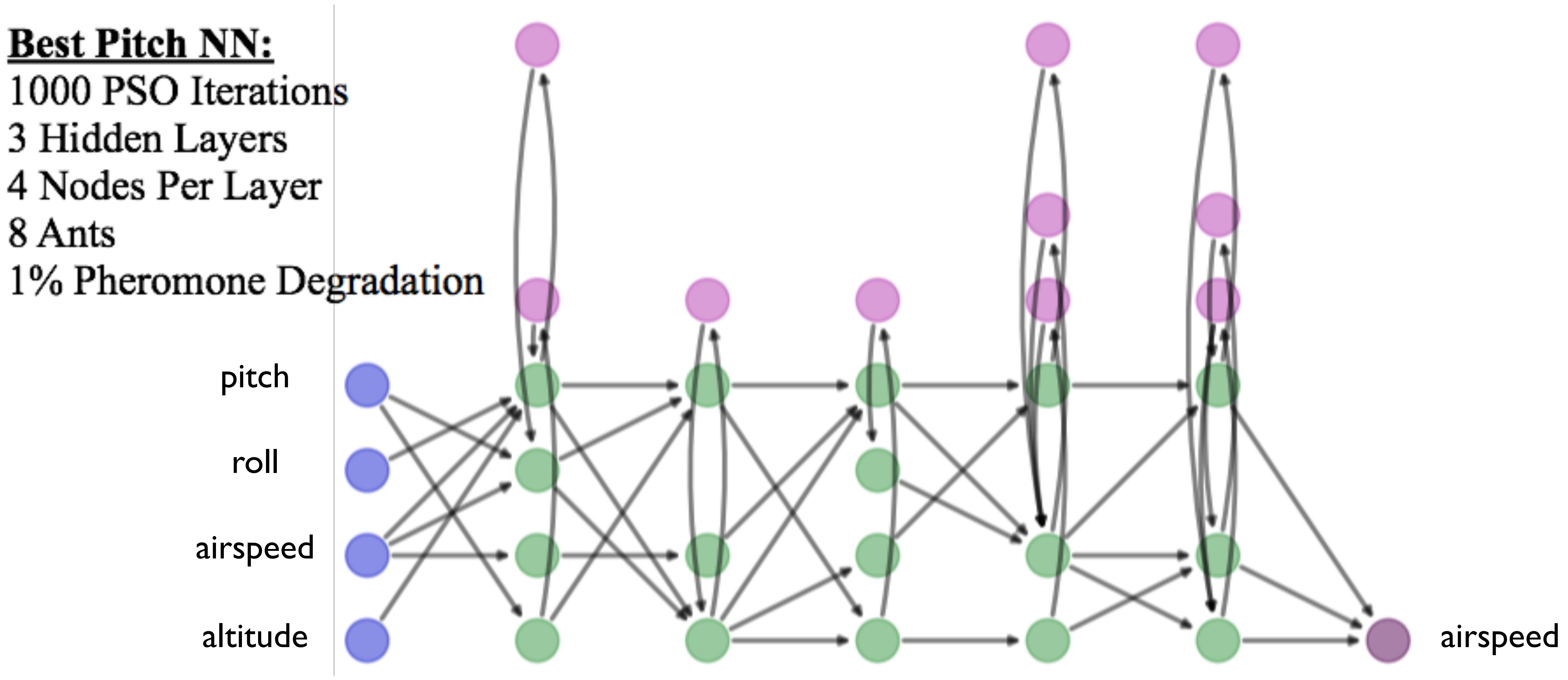
Results: Best Found Neural Networks

The following shows the best neural network for predicting pitch as evolved by the ACO strategy.



Results: Best Found Neural Networks

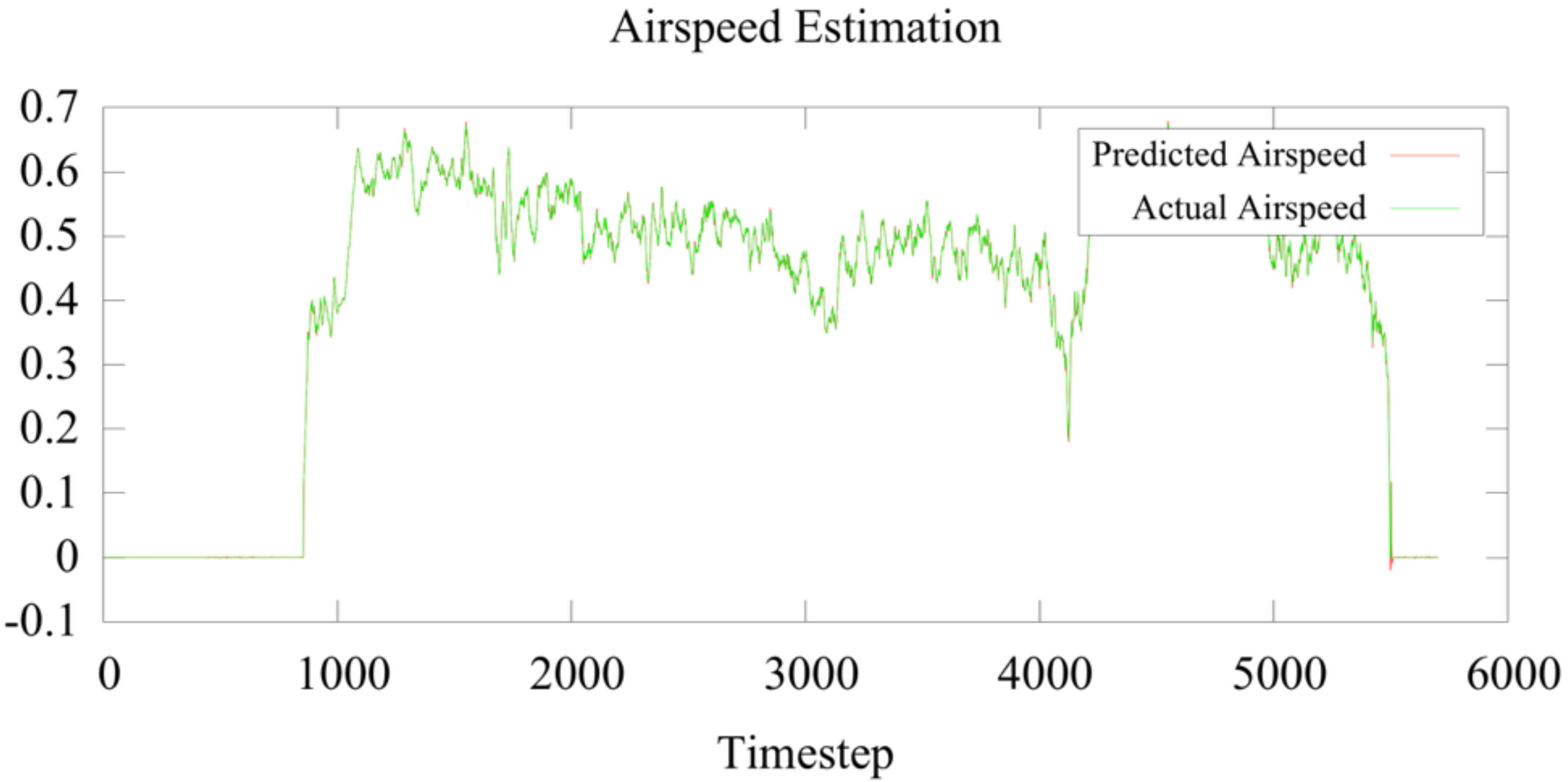
The following shows the best neural network for predicting roll as evolved by the ACO strategy.



Best Pitch NN:
1000 PSO Iterations
3 Hidden Layers
4 Nodes Per Layer
8 Ants
1% Pheromone Degradation

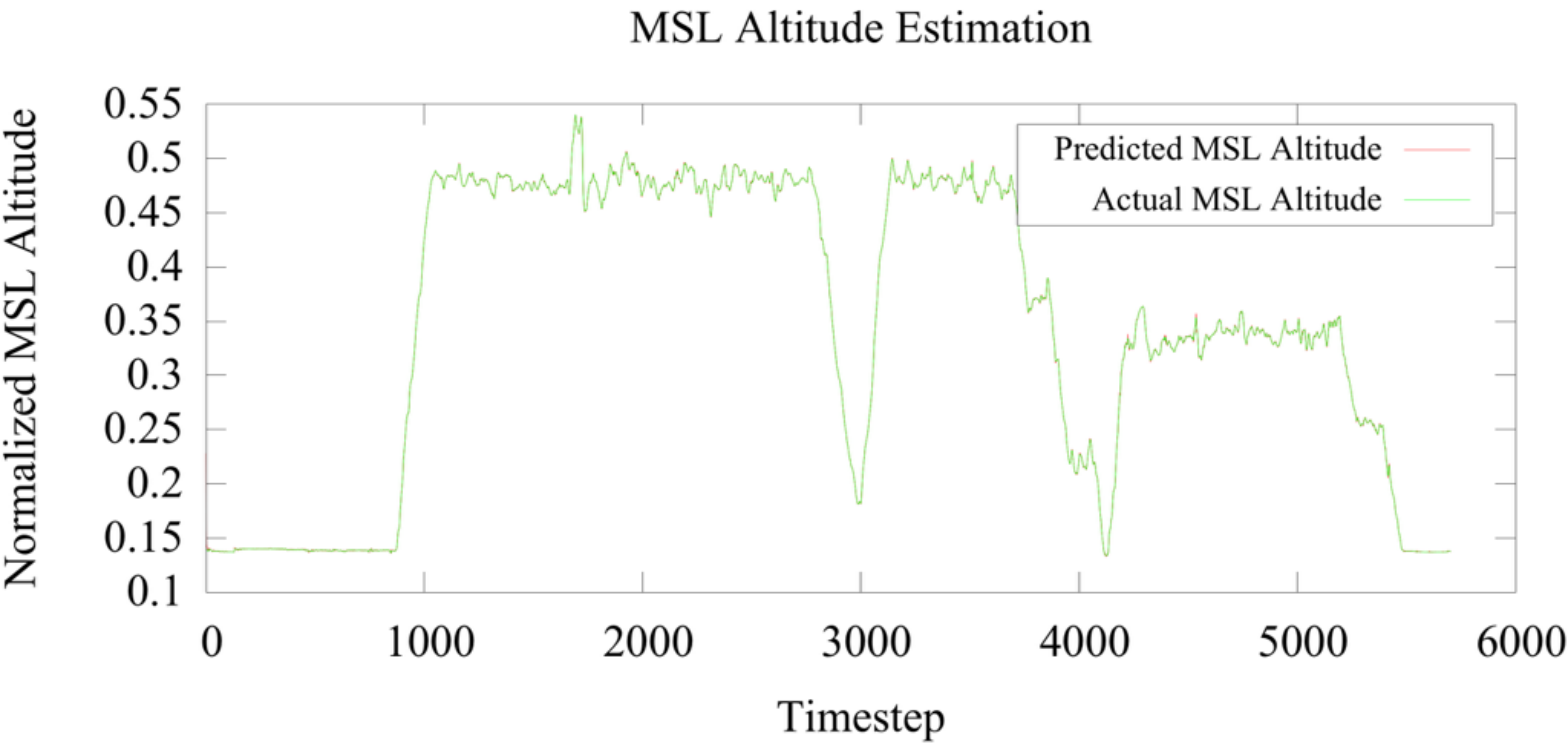
Results: Predictions

Predictions of the best RNN evolved and trained on flight 13588 used to predict flight 17269 for airspeed.



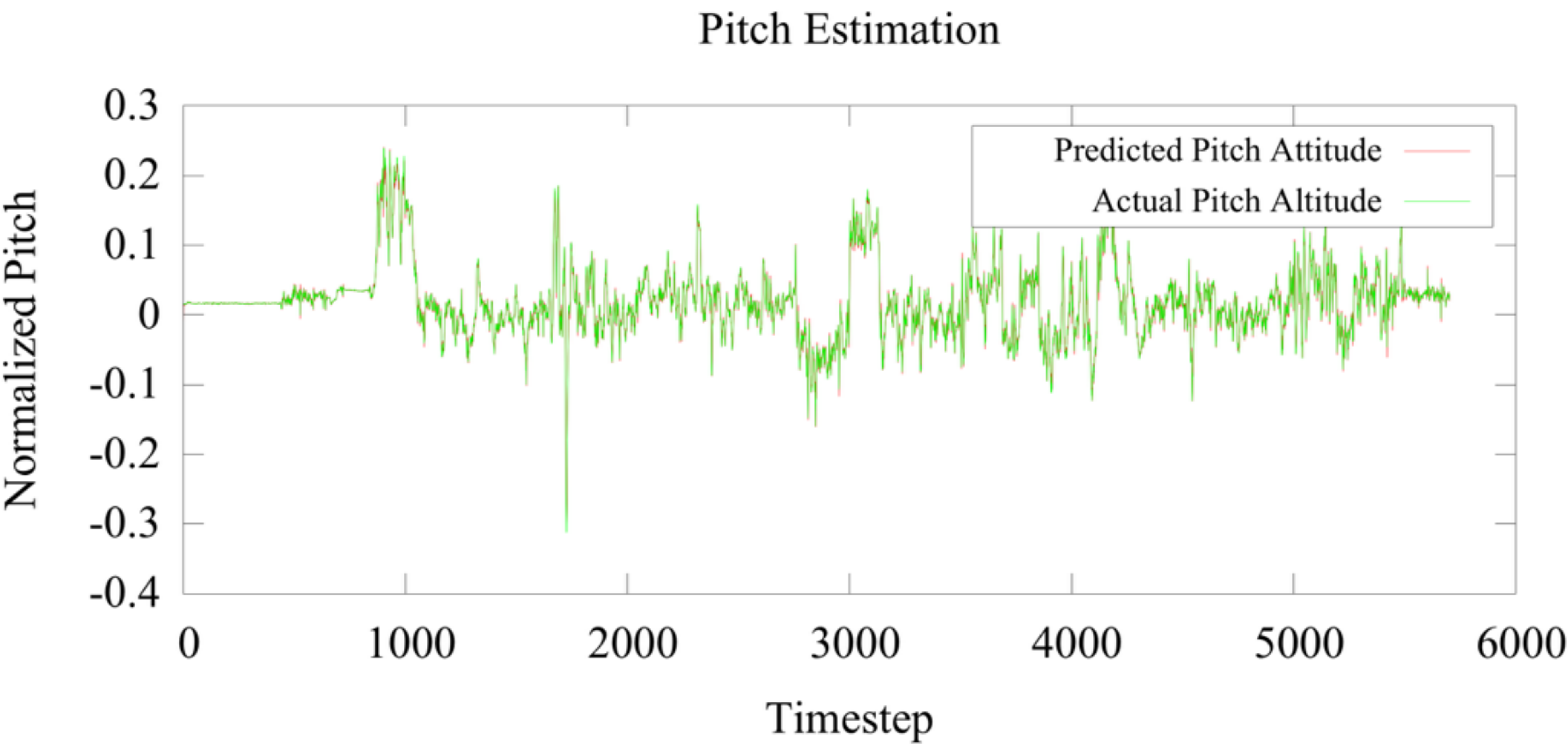
Results: Predictions

Predictions of the best RNN evolved and trained on flight 13588 used to predict flight 17269 for altitude.



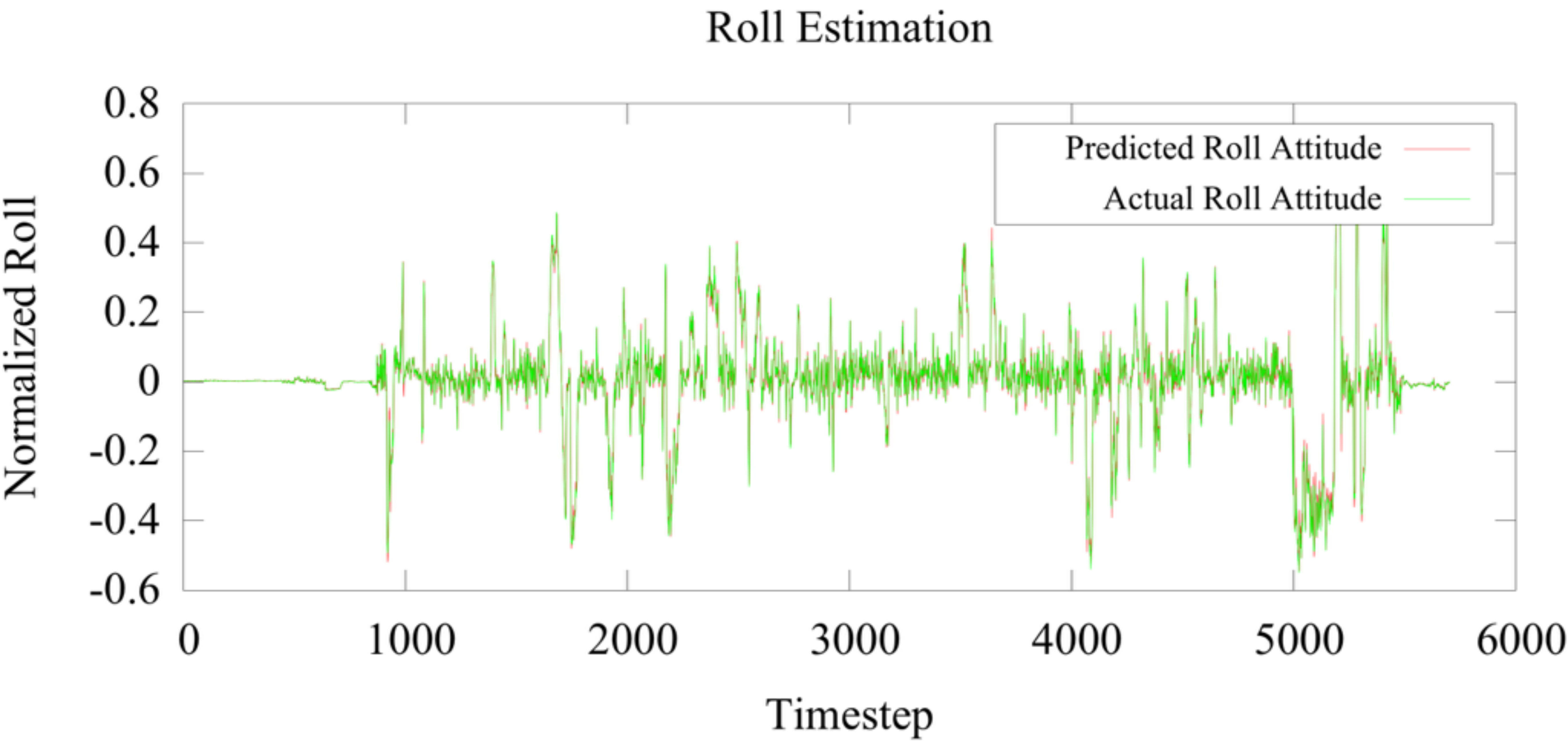
Results: Predictions

Predictions of the best RNN evolved and trained on flight 13588 used to predict flight 17269 for pitch.



Results: Predictions

Predictions of the best RNN evolved and trained on flight 13588 used to predict flight 17269 for pitch.



Results: Comparison to Prior Results

Airspeed

Method	13588	15438	17269	175755	24335
$t_{i+1} = t_i$	0.00512158	0.00316859	0.00675531	0.00508229	0.00575537
Prior Best	0.00472131	0.00250284	0.00656991	0.00465581	0.00495454
Best ACO	0.00279963	0.00145748	0.00433578	0.0028908	0.00305361

Altitude

Method	13588	15438	17269	175755	24335
$t_{i+1} = t_i$	0.00138854	0.00107117	0.00200011	0.00137109	0.00192345
Prior Best	0.000367535	0.000305193	0.000895711	0.000399587	0.000485329
Best ACO	0.0002183	0.000160932	0.000353502	0.000224827	0.000249197

Pitch

Method	13588	15438	17269	175755	24335
$t_{i+1} = t_i$	0.0153181	0.010955	0.0148046	0.0161251	0.0173269
Prior Best	0.014918	0.0100763	0.0147712	0.01514	0.0160249
Best ACO	0.00606664	0.00498241	0.00837594	0.005864	0.00733882

Roll

Method	13588	15438	17269	175755	24335
$t_{i+1} = t_i$	0.0158853	0.00604479	0.0204441	0.012877	0.0192648
Prior Best	0.0154541	0.00587058	0.0206536	0.0127999	0.0182611
Best ACO	0.0155934	0.00900393	0.0237235	0.0151416	0.0200261

These neural networks were compared to a random noise estimator ($t_{i+1} = t_i$), best results from our previous publication, and were also run on four other flights, IDs 15438, 17269, 175755 and 24335.

On average compared to previous best results, the ACO evolved neural networks provided a 63% improvement over airspeed, a 97% improvement over altitude and a 120% improvement over pitch, without requiring additional input neurons from previous timesteps (which the previous strategy allowed). Roll however, did not improve (maybe needed to train th RNNs longer).

Conclusions & Future Work

Conclusions

Ant colony optimization can be used for neuro-evolution. Our strategy is easily parallelizable and can use any neural network training algorithm.

Evolutionary algorithms can be well applied to training deep recurrent neural networks.

Data used in this work has been made available as the first data release from the NGAFID for other researchers to compare (and hopefully beat!) our results.

Future Work

Compare ACO to modified NEAT/Hyper-NEAT.

Investigate different strategies for incrementing/decrementing pheromones.

Can we apply ACO to evolving large scale NNs for computer vision tasks? Can they also select different neuron types (ReLU units, max pooling units, etc).

Can we go deeper in terms of recurrency? What if memory neurons had their own memory neurons, and so on. Can we also utilize LSTM memory neurons?

Questions?

tdesell@cs.und.edu

http://people.cs.und.edu/~tdesell

https://github.com/travisdesell/tao

http://people.cs.und.edu/~tdesell/ngafid_releases.php