Competitive Two-Island Cooperative Co-evolution for Training Feedforward Neural Networks for Pattern Classification Problems

Rohitash Chandra * † and Gary Wong * †
* School of Computing Information and Mathematical Sciences
University of the South Pacific, Suva, Fiji. http://scims.fste.usp.ac.fj/
† Artificial Intelligence and Cybernetics Research Group, Software Foundation,
Nausori, Fiji. http://aicrg.softwarefoundationfiji.org/
Email: c.rohitash@gmail.com, Email: gary.wong.fiji@gmail.com

***Abstract*—In the application of cooperative coevolution for neuro-evolution, problem decomposition methods rely on architectural properties of the neural network to divide it into subcomponents. During every stage of the evolutionary process, different problem decomposition methods yield unique characteristics that may be useful in an environment that enables solution sharing. In this paper, we implement a two-island competition environment in cooperative coevolution based neuro-evolution for feedforward neural networks for pattern classification problems. In particular the combinations of three problem decomposition methods that are based on the architectural properties that refers to neural level, network level and layer level decomposition. The experimental results show that the performance of the competition method is better than that of the standalone problem decomposition cooperative neuro-evolution methods.**

I. Introduction

In nature, it is difficult for an organism to acquire resources that are concurrently being consumed or defended by competing species [1]. In a way, competition reduces each others growth and reproductive processes but in turn ultimately enhances survival characteristics which are found in the genes of competing species [2]. Collaboration enables survival amongst species in environments with limited resources [1]. The Reproduction and creation of future populations are also moderately effected by environmental conditions [2].

In evolutionary algorithms, early methods of competition have been incorporated using two populations to represent ‘parasites’ and ‘hosts’ with evolutionary mechanisms such as fitness sharing, elitism and selection [3].

Cooperative Coevolution (CC) divides a problem into subcomponents and employs evolutionary algorithms to collectively solve the main problem [4]. Each subcomponent is a partial solution to the original problem and is conjoined with others through evolutionary processes where a more efficient solution may be obtained [5]. Classic cooperative co-evolutionary method appeals to problems that are fully separable [6] where no or little interaction is present amongst decision variables [7]. Cooperative coevolution has mainly been applied for large scale optimisation problems [6], [8], multi-objective problems [9] and neuro-evolution of feedforward and recurrent neural networks for problems that involve pattern classification [10], [11], [12], [13], control [14], [15] and time series prediction [16], [17], [18].

In the case of neuro-evolution using cooperative coevolution, the issue of inter-dependencies between decision variables or synapses have also been explored however, more work needs to be done as it is difficult to fully decompose the neural network as the problem at hand also plays a part [19]. Problem decomposition at the synapse level creates a subcomponent for every synapse which has also showed to be more effective for control problems [14], [15] and time series prediction problems [16] while less effective for pattern classification problems [12], [19].

Another problem decomposition method is at neuron level which groups synapses that are attached to a particular neuron and has shown the best results in pattern classification [12], [20] and competitive results with synapse level decomposition for time series prediction problems [16]. Both decomposition methods have good levels of success and unique characteristics for the varying problems and neural network architectures. The neural network architecture (feedforward vs recurrent) the training problem type (control, pattern classification or time series prediction) are two important aspects that need to be taken into account when cooperative coevolution is used for neuro-evolution [19].

Competition in cooperative coevolution has been used in multi-objective [21] and dynamic environment optimizations where problem decomposition methods adapt to changing environments [22]. Additionally, the methods are routinely swapped through adaptation over the course of an evolutionary phase. Although the approach of adaptation has shown to be resourceful on pattern recognition [23] and grammatical inference problems [24], there is a disadvantage in terms of cost with regards to parameter tuning that determines when and how to adapt problem decomposition methods [24]. Generally, these settings refer to the estimated times at which to switch decomposition methods and the duration of use for each method [23].

In this paper, we apply competitive island-based cooperative coevolution (CICC) to pattern classification problems that uses varying decomposition methods to enable competition and collaboration. In CICC, problem decomposition methods are implemented as islands that compete and collaborate with each other using injection methods. In our previous work, CICC has shown to yield promising results when applied to time series prediction problems [17].

The main contribution of this paper is to explore if CICC is effective on pattern classification problems. The measure of effectiveness here is the ability of the algorithm to converge faster and also to lessen the number of function evaluations while using different problem decomposition methods simultaneously.

The remaining sections of the paper are structured as follows. Section 2 provides a concise background on cooperative coevolution in neuro-evolution. Section 3 provides information on competition and collaboration in feed-forward neural networks. Section 4 details the experiment configuration, results and discussion. Section 5 will conclude with keynotes on future work.

II. Background

A. Cooperative Coevolution for Neuro-Evolution

A conventional way of applying a cooperative co-evolutionary algorithm (CCEA) to a problem begins by dividing it into smaller groups or subcomponents. When a subcomponent is created, it is implemented as a sub-population in CCEA and subjected to a predefined evolutionary algorithm (EA) [25], [17], [4]. The cooperative factor here relates to the relationships among these components when individuals are evaluated though shared fitness evaluation. The process of breaking a problem down into subcomponents is called problem decomposition where the size and encoding of each subcomponent grossly depends on the problem at hand [19].

Original cooperative co-evolutionary techniques have been used for general function optimization problems where each dimension (decision variable) is implemented as a separate subcomponent [4]. Although later studies have found that the original scheme appeals solely to fully separable problems [6], there have been continuous work on applying cooperative coevolution to large scale non-separable function optimization problems [6], [26], [27], [8]. The separability of a function with m variables is defined by whether or not it can be expressed as a sum of m functions in relation to a single variable [28]. Where a problem is non-separable there exists interdependencies among the variables when compared to non-separable problems where there is little to none. Moreover, real world problems can either be fully separable or fully non-separable.

In cooperative co-evolution, sub-populations incorporate a round-robin selection procedure during evolution for a certain number of generations which is referred to as the depth of search (predetermined according to problem nature). The depth of search is used to assess the ability of problem decomposition methods to group interacting variables into separate components [20].

III. Competition and Collaboration in Cooperative Coevolution

In an environment with multiple species, competition can be seen as a means of 'natural rivalry' for access to limited resources [1], [2]. Individual species compete for these resources that may vary according to the habitat, environmental conditions and external sources such as human interaction [2]. Collaboration on the other hand is also an important feature used for survival in nature. With collaboration, species with different adaptation characteristics share resources when faced with specific challenges. These individual species or subcomponents are implemented as sub-populations in a cooperative co-evolutionary framework where genetic materials are not shared with other sub-populations. These genetic materials in sub-populations will not be shared in conventional cooperative coevolution but can only be shared through collaboration which is helpful in evolutionary procedures [17], [17]. In general, competition and collaboration are vital components of evolution where different groups of species compete for resources in the same environment. In cooperative coevolution, the variety of species are represented as problem decomposition methods [12]. These decomposition methods participate in competition and collaboration through fitness evaluation during evolution.

In this section, we apply a cooperative co-evolutionary method called *Competitive Island-Based Cooperative Coevolution (CICC)* for pattern classification. CICC employs different problem decomposition methods that compete with different features in terms of diversity and degree of non-separability [19]. The method employs the strength of different problem decomposition methods which are described by the level of interaction between variables and the diversity (total number of sub-populations) during evolution [19].

In the remaining parts of the discussion, the different types of problem decomposition methods will be referred to as '*islands*'. In CICC, the different islands compare solutions after fixed intervals of time (island evolution time) and exchange the more promising solution between them. For the case of this model, two islands are used which are derived from the individual decomposition methods. The details of each island (decomposition methods) are given below.

1) **Network Level**: Standard neuro-evolution where the entire network is used 'as-is' without decomposition.
2) **Neural Level problem decomposition**: Decomposes the network into neuron level. The number of neurons in the hidden and output layer determines the number of subcomponents [19], [20].
3) **Layer Level problem decomposition**: Decomposes the network by layers into two parts. The first subcomponent contains all the weights from input to hidden layer and the second subcomponent contains all the weights from hidden to output layer.

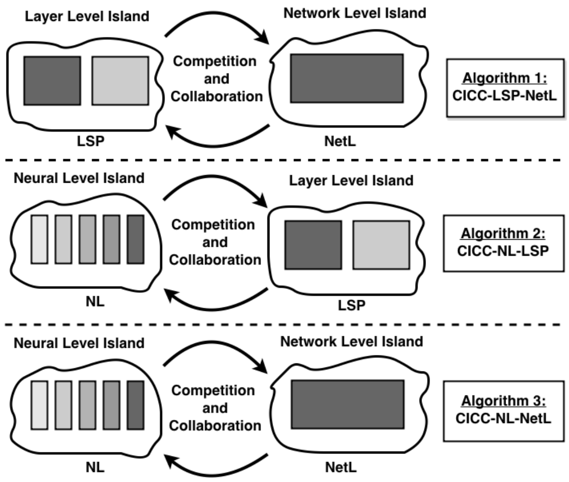

Fig. 1. Two-Island CICC for Neuro-Evolution. The CICC methods employ neural level (NL) vs network level (NetL), neural level vs layer level (LSP) and layer level vs network level island based competition.

Alg. 1 CICC for Feedforward Neural Network (LSP vs NetL)

Stage 1: Initialisation:

i. Cooperatively evaluate Layer Level island
ii. Equalize sub-populations on both islands
iii. Evaluate Network Level island

Stage 2: Evolution:

while *FuncEval* $\leq$ *GlobalEvolutionTime* **do**
  while *FuncEval* $\leq$ *Island-Evolution-Time* **do**
    foreach *Sub-population at Layer Level Island* **do**
      foreach *Depth of* n *Generations* **do**
        Create new individuals using genetic operators
        Cooperative Evaluation
      end
    end
  end
  while *FuncEval* $\leq$ *Island-Evolution-Time* **do**
    for the Sub-population at Network Level Island
    foreach *Depth of* n *Generations* **do**
      Create new individuals using genetic operators
      Cooperative Evaluation
    end
  end
  Stage 3: Competition: Compare and mark the island with best fitness.
  Stage 4: Collaboration: Inject the best individual from the island with better fitness into the other island.
  if *Layer Level Island*$\leq$ *Network Level Island* **then**
    Copy Layer Level Island best into chosen Network Level Island Individual.
  end
  else
    Copy Network Level Island best into chosen Layer Level Island Individuals.
  end
end

The CICC methodology for two-island competition is given in Algorithms 1, 2 and 3.

For all algorithms; In Stage 1, the sub-populations are initialised and cooperatively evaluated using network, neuron and layer level problem decomposition methods. Before evaluating the network level island, the sub-populations from neuron level

Alg. 2 CICC for Feedforward Neural Network (NL vs LSP)

Stage 1: Initialisation:

i. Cooperatively evaluate Neural Level island
ii. Equalize sub-populations on both Method 3:islands
iii. Evaluate Layer Level island

Stage 2: Evolution:

while *FuncEval* $\leq$ *GlobalEvolutionTime* **do**
  while *FuncEval* $\leq$ *Island-Evolution-Time* **do**
    foreach *Sub-population at Neural Level Island* **do**
      foreach *Depth of* n *Generations* **do**
        Create new individuals using genetic operators
        Cooperative Evaluation
      end
    end
  end
  while *FuncEval* $\leq$ *Island-Evolution-Time* **do**
    foreach *Sub-population at Layer Level Island* **do**
      foreach *Depth of* n *Generations* **do**
        Create new individuals using genetic operators
        Cooperative Evaluation
      end
    end
  end
  Stage 3: Competition: Compare and mark the island with best fitness.
  Stage 4: Collaboration: Inject the best individual from the island with better fitness into the other island.
  if *Neural Level Island*$\leq$ *Layer Level Island* **then**
    Copy Neural Level Island best into chosen Layer Level Island Individual.
  end
  else
    Copy Layer Level Island best into chosen Neural Level Island Individuals.
  end
end

island are copied to ensure that both islands start from the same position in search space thus creating an equal and fair competition field. This also ensures that the material injected from the winner island is valid and applicable as the neural network training problem is multi-modal.

In Stage 2, the islands are exposed to evolution in an island based round-robin fashion where each island is evolved for a predefined set amount of time based on predefined fitness evaluation. The time taken to evolve an island is called *'island evolution time'* and is given by the number of cycles that formulates the required number of function evaluations in the respective islands [17].

In competition phase (Stage 3), the best solutions from all the islands are compared and the overall best marked for injection into other islands. The best solution is made up of the best individuals from all the sub-populations.

In the collaboration mechanism (Stage 4), the algorithm needs to take into account how the solution from one island will be transferred into the rest of the islands. As presented in previous work [17], the respective islands need to be given the same number of function evaluations. Due to the requirement that each island be evaluated for complete cycles, the number of function evaluations for both islands should not be exactly the same but rather close approximates of each other.

With reference to cooperative co-evolution, a cycle is defined by the evolution time of the sub-populations when

Alg. 3 CICC for Feedforward Neural Network (NL vs NetL)

Stage 1: Initialisation:

i. Cooperatively evaluate Neural Level island
ii. Equalize sub-populations on both islands
iii. Evaluate Network Level island

Stage 2: Evolution:

```
while FuncEval ≤ GlobalEvolutionTime do
    while FuncEval ≤ Island-Evolution-Time do
        foreach Sub-population at Neural Level Island do
            foreach  Depth of n Generations do
                Create new individuals using genetic operators
                Cooperative Evaluation
            end
        end
    end
    while FuncEval ≤ Island-Evolution-Time do
        for the Sub-population at Network Level Island
        foreach  Depth of n Generations do
            Create new individuals using genetic operators
            Cooperative Evaluation
        end
    end
    Stage 3: Competition: Compare and mark the island with best fitness.
    Stage 4: Collaboration: Inject the best individual from the island with
    better fitness into the other island.
    if  Neural Level Island≤ Network Level Island then
        Copy Neural Level Island best into chosen Network Level Island
        Individual.
    end
    else
        Copy Network Level Island best into chosen Neural Level Island
        Individuals.
    end
end
```

evolved for t number of generations in a round-robin fashion. Once the participating islands have been evolved for the island evolution time, the algorithm checks and compares the best solution of all these islands. The top contending solution among the islands is copied among them, the reason for this is to help the rest of the islands in the next phase of competition.

Individuals in the respective sub-populations are cooperatively evaluated by concatenating a chosen individual from a given sub-population SP with the best individuals from other sub-populations [4], [12], [20], [16]. The chosen individual is encoded into the feed-forward neural network where the fitness can be computed. The goal of evolution is to increase fitness while decreasing the network error. The reason for this is to ensure that each sub-component in the network is evaluated till cycle completion.

A. Competition

In competition, each island employs a distinct problem decomposition method. The number of function evaluations per island is dependent on the the number of sub-populations used by the decomposition method. There are more sub-populations in the neural level island then the two sub-populations of the layer level island and single population of the network level island.

Moreover, each island needs to have the same amount of evolution time with the same or similar required number of function evaluations. A complete cycle for an island is distinct from the rest of the islands thus the function evaluation count for each cannot be exactly the same but may have similar evaluation times with a slight variance in the degree of difference.

B. Collaboration

The comparison of best individuals between islands do not occur until the commence of the collaborative process. The island that contains an individual with better solution is copied to the other islands as shown in Stage 4 of Algorithms 1, 2 and 3 and Figure 2. The way in which an individual is concatenated into another island is paramount since the size and number of sub-populations vary from island to island. When the winning island copies or injects its best individual to the other island, they are concatenated and then later broken down and mapped into the other island, keeping into account that the size of the subcomponents will not be the same, e.g, when best individuals layer level sub-populations island are injected in neural level island.

After the injection, the copied individuals can be re-evaluated. This re-evaluation strategy can be omitted if the fitness is transferred along with the individual in order to save function evaluation time. Further on, it is important that this fitness and solution be injected in the new island(s) at the exact position the best individual was derived from the original winner island.

Additionally, it is worth noting that each of these individuals have distinct fitness values. We take the best fitness value of the last sub-population in the winner island and replace the old fitness values on the other islands (sub-populations) with this new fitness.

C. Island Initialisation

We initialise all the islands in Stage 1 of Algorithm 1 with random real numbers and ensure that all the islands have the same values or beginning position in search space. Before the sub-populations on the islands are evaluated, we initialise all of them by copying the population from one island to the rest of the participating islands. The reason for doing this is to create a fair competition where all the participating problem decomposition methods start from the same point and have a single local minimum at the beginning of evolution. We note that the search space of neural network optimisation is multi-modal and there can be several unique solutions with equal value or fitness. Therefore it is important for islands that compete and collaborate with each other to have similar search space from the same region so that island injection can be better utilised.

D. Performance Evaluation

The optimisation time for CICC is computed from two empirical scalar quantities which are the number of function evaluations and the total success rate [17]. These two measurements are also used as the stopping criteria for an experimental run. Further on, the success rate is used to determine if the algorithm can output a desirable solution within a specified

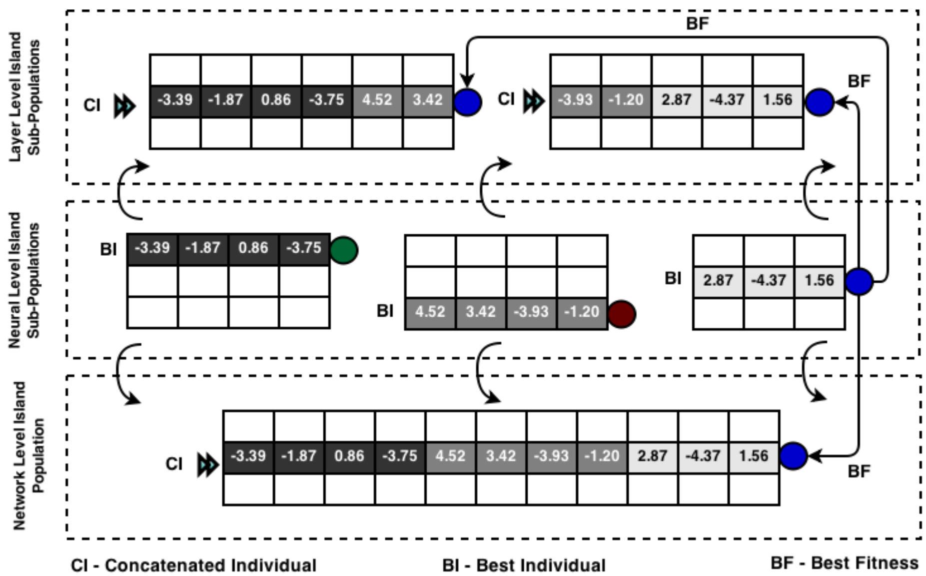

Fig. 2. Concatenating the best individuals from neural level island and injecting into network level island. The same is done from neural level to layer level island for two-island competition. Note the fitness of the concatenated individual is acquired from the fitness of the last best individual from the neural level island. The position of the concatenated individual also comes from here. In CICC-NL-NetL, the transfer of best individuals from network level to neural level works exactly in the same manner except, the network level individual is broken down to subcomponents to match neural level. In CICC-NL-LSP, transferring best individuals requires careful attention as the receiving island component size may differ. The figure shows that the fitness of the single best individual from the neural level replaces the fitness of all best individuals on the second island.

time. Where an experiment run yields a desired solution, only then will it be considered a success. The training percentage of an experiment run is used to compare to the desired success rate and is important for robustness [17]. Additionally, a successful run may be awarded provided the max function evaluation time has not been reached. The desired solution during training of the neural network is defined by a minimum network error or classification performance on the training data set.

In this study, the algorithm is tested using different counts of hidden neurons to analyse the appropriate network configuration for scalability and robustness. The optimisation time is obtained from the average function evaluation count in n experiments. Note, experiments that do not converge within the specified evaluation period is also included in the computation of the optimization time.

TABLE I
DATASET INFORMATION AND NEURAL NETWORK CONFIGURATION

Problem	Input	Output	Min. Train (%)	Max. Time	Samples
4-Bit	4	1	–	30000	16
Wine	13	3	95	15000	178
Iris	4	3	95	15000	150
Heart	13	1	88	50000	303
Cancer	9	1	95	15000	699

IV. Simulation and Analysis

In this section we present an experimental study of *Competitive Island-Based Cooperative Co-evolution* (CICC). The cooperative co-evolutionary approach used here has 200 individuals in each sub-population. The chromosomes in the sub-

TABLE IV
PERFORMANCE FOR THE **HEART** PROBLEM

		Heart				
Method	**H**	($\bar{x}$)		**Test**	**Error**	**(%)**
CC-NL	6	19404	7983	78.61	1.62	80
	8	**15719**	**3509**	79.88	1.08	100
	10	**35760**	**7818**	80.00	3.2	50
	12	24445	4202	80.55	2.13	100
	14	18360	3500	78.77	1.35	100
CC-NetL	6	41958	7839	79.62	1.56	30
	8	**39240**	**8831**	80.88	0.72	50
	10	42480	6489	80.92	1.19	60
	12	**46210**	**8631**	77.23	1.45	40
	14	45312	8871	74.12	1.67	10
CC-LSP	6	37908	3954	81.48	0.59	30
	8	**36288**	**3674**	81.84	0.90	40
	10	46890	4735	80.55	2.30	20
	12	48168	5617	82.22	1.53	20
	14	**49644**	**1405**	80.00	2.80	10
CICC-NL-NetL	6	**31038**	**6321**	79.44	1.97	60
	8	**13824**	**2780**	79.11	1.26	100
	10	18414	2007	78.55	1.57	100
	12	20358	4116	79.55	1.11	100
	14	19830	3287	80.44	1.38	100
CICC-NL-LSP	6	**27349**	**9678**	78.61	1.57	80
	8	22356	8686	80.00	1.19	90
	10	20922	3923	80.33	1.79	100
	12	**18564**	**3537**	78.77	2.51	100
	14	21690	2849	79.33	1.20	100
CICC-LSP-NetL	6	33948	6875	79.44	1.48	60
	8	**31234**	**6234**	80.44	1.13	60
	10	**38964**	**7689**	79.62	0.90	70
	12	35214	6941	78.34	0.98	40
	14	32015	6671	80.12	1.10	50

H = Hidden Neurons (%) = Percentage Success Rate
($\bar{\mathbf{x}}$) = Mean Function Evaluations

TABLE II
PERFORMANCE FOR THE **WINE** AND **4-BIT** PROBLEMS

Method	H	Wine ($\bar{x}$)		Test	Error	(%)	H	4-Bit ($\bar{x}$)		Test	Error	(%)
CC-NL	4	**5611**	**508**	94.20	1.14	100	4	**11151**	**6237**	100.00	-	80
	6	6068	482	94.30	0.84	100	6	6001	1921	100.00	-	100
	8	6515	530	93.10	1.03	100	8	**5772**	**1093**	100.00	-	100
	10	7238	991	92.60	0.98	100	10	7012	2314	100.00	-	100
	12	**7698**	**864**	92.85	1.10	100	12	6318	862	100.00	-	100
CC-NetL	4	**13314**	**1430**	94.38	3.18	40	4	**25819**	**5497**	100.00	-	20
	6	14539	1094	90.83	1.33	30	6	23004	6667	100.00	-	30
	8	14641	834	92.50	1.46	10	8	12614	5549	100.00	-	80
	10	**14985**	**864**	91.23	1.64	10	10	**8100**	**1028**	100.00	-	100
	12	14652	712	92.31	1.39	10	12	9590	1147	100.00	-	100
CC-LSP	4	**10724**	**1348**	93.00	1.58	67	4	**29472**	**1428**	72.50	-	10
	6	11356	1149	94.17	1.46	80	6	20664	7278	87.50	-	40
	8	13855	1149	93.60	2.2	53	8	21696	5309	97.50	-	60
	10	**15686**	**822**	92.92	4.23	20	10	**8040**	**1331**	100.00	-	100
	12	15434	745	91.20	3.56	10	12	9360	2044	100.00	-	100
CICC-NL-NetL	4	**5950**	**566**	93.75	1.38	100	4	**12090**	**5821**	100.00	-	80
	6	6696	638	93.58	1.68	100	6	9198	4577	100.00	-	90
	8	6820	423	94.08	1.14	100	8	**5454**	**927**	100.00	-	100
	10	8424	1685	94.00	1.85	30	10	9666	947	100.00	-	100
	12	**8820**	**1592**	93.25	2.08	30	12	7098	903	100.00	-	100
CICC-NL-LSP	4	**6174**	**687**	94.16	1.46	100	4	**17423**	**6736**	100.00	-	60
	6	6750	628	95.16	0.76	100	6	4074	931	100.00	-	100
	8	7128	657	94.83	1.32	100	8	**2808**	**360**	100.00	-	100
	10	8164	601	93.66	1.30	100	10	12246	5877	100.00	-	80
	12	**8670**	**893**	93.36	1.07	96	12	3588	870	100.00	-	100
CICC-LSP-NetL	4	8004	2274	94.23	1.42	90	4	**21563**	**10231**	100.00	-	60
	6	**5585**	**1018**	94.23	1.24	100	6	14400	6530	100.00	-	80
	8	7764	1866	94.60	1.26	90	8	6948	4875	100.00	-	90
	10	12600	1676	93.12	2.14	50	10	7116	4021	100.00	-	100
	12	**12876**	**1723**	94.52	0.86	60	12	**6816**	**4860**	100.00	-	90

H = Hidden Neurons (%) = Percentage Success Rate ($\bar{x}$) = Mean Function Evaluations

TABLE III
PERFORMANCE FOR THE **IRIS** AND **CANCER** PROBLEMS

Method	H	Iris ($\bar{x}$)		Test	Error	(%)	H	Cancer ($\bar{x}$)		Test	Error	(%)
CC-NL	4	**5112**	**1750**	96.50	1.07	100	4	**4122**	**1156**	96.73	0.49	100
	6	4080	1570	95.50	1.63	100	6	4930	1566	97.13	0.57	100
	8	4392	1370	97.00	0.87	100	8	4811	1024	97.77	0.37	100
	10	4492	1135	96.00	1.75	100	10	4963	1315	98.16	0.34	100
	12	**3376**	**1760**	95.00	2.4	100	12	**5116**	**962**	97.77	0.48	100
CC-NetL	4	**5950**	**3567**	95.00	2.77	100	4	**8448**	**2869**	95.54	0.98	80
	6	10836	2705	97.50	1.37	100	6	10188	2802	95.89	0.47	70
	8	12243	2262	96.66	1.33	60	8	10848	3527	95.69	0.27	50
	10	**13546**	**3136**	92.55	1.32	20	10	**11220**	**2927**	95.94	0.38	50
	12	12138	3859	96.66	1.33	60	12	9000	3105	96.04	0.71	70
CC-LSP	4	**5614**	**3674**	94.50	2.61	100	4	9820	3203	95.89	0.43	70
	6	7017	3768	94.38	3.18	80	6	**6336**	**3173**	96.23	0.43	90
	8	**14637**	**479**	95.00	2.2	40	8	8025	2604	96.04	0.71	80
	10	12609	2964	93.75	4.23	40	10	11220	2644	96.43	0.47	60
	12	13671	4321	91.02	4.54	20	12	**13305**	**2114**	96.28	0.49	40
CICC-NL-NetL	4	**4248**	**1441**	97.00	0.87	100	4	**8010**	**2903**	97.22	0.28	90
	6	5280	1862	95.00	2.74	100	6	4410	738	97.52	0.51	100
	8	5640	1868	94.31	1.84	100	8	4806	1889	97.77	0.44	100
	10	4392	1253	96.50	1.76	100	10	**4290**	**780**	97.87	0.41	100
	12	**6384**	**2085**	96.00	1.02	100	12	4602	401	98.61	0.30	100
CICC-NL-LSP	4	**5656**	**2520**	95.00	1.65	100	4	**6150**	**3431**	96.58	0.37	80
	6	7440	2515	93.50	1.77	100	6	4368	1243	97.27	0.31	100
	8	7320	2148	94.25	1.98	100	8	**3456**	**522**	97.57	0.32	100
	10	7440	2198	95.75	1.47	100	10	4092	542	98.21	0.39	100
	12	**7764**	**2156**	96.00	0.60	100	12	4368	618	98.21	0.36	100
CICC-LSP-NetL	4	**4044**	**2324**	96.25	1.03	100	4	5133	1242	95.93	0.26	88
	6	6028	2615	94.50	1.51	100	6	**6405**	**1475**	96.26	0.27	82
	8	6668	2490	96.00	1.02	100	8	4874	1269	96.38	0.29	88
	10	7024	2809	96.00	1.02	100	10	**3280**	**928**	96.38	0.21	96
	12	**7724**	**2611**	94.50	2.38	100	12	3288	1018	96.57	0.23	94

H = Hidden Neurons (%) = Percentage Success Rate ($\bar{x}$) = Mean Function Evaluations populations are initialised with random data in the range of [-5, 5] as shown in Figure 2.

A. Classification Problems and Configuration

To simulate the CICC architecture in our previous work [29], we use four pattern classification problems from the UCI Machine Learning Repository [30] which are Wine, Iris, Cleveland Heart Disease and Wisconsin Breast Cancer. The only problem that is not derived from the repository is the 4-bit parity problem. In this problem, the even parity is computed based on the even count of 1's in the input data. These problems have been frequently used in other studies to evaluate performances of new methods [12], [19]. According to the contributor of the problem set, the Wisconsin Breast Cancer problem has 16 missing values and is class unbalanced (34.5% Malignant and 65.5% Benign) [30].

Further details of each problem are provided in Table I. In the 4-Bit parity problem, there is no maximum training time for the network but rather it is trained until the mean-squared error goes below 1E-3 [19]. Apart from the 4-Bit parity problem, 30% of the data is used for testing and 70% for training. In each problem, at each hidden neuron (e.g. 4, 6, 8, 10), there are 50 independent runs initialised with different positions in the search spaces.

With reference to neural network topology, the number of dimensions in the optimisation problem is determined by the number of problems tested. This is used to gauge the performance of the CICC method on various levels of scalability, robustness and difficulty. In the standalone methods used (neural level (NL), network level (NetL) and layer level(LSP)), the termination condition for each problem is provided in Table I as maximum time (*Max. Time*).

B. Results and Discussion

The CICC methods employed in this study have shown better performance when compared to the standalone co-evolutionary approaches at the neural, layer and network level. This improvement in performance can be attributed to the collaborative features of two-island competition and the implementation of islands on separate threads. A major goal of employing CICC is to improve experiment success rates while lowering the mean function evaluations. Where an experiment has a max time of k evaluations, the max time permitted overall for both islands would be $2k$, however, per each island, the thread is k evaluations.

The results of the experiments are given in Tables II - IV where a comparison is made between standalone cooperative co-evolutionary techniques and the CICC methods. Note that all the methods incorporated G3-PCX evolutionary algorithm in their sub-populations.

The results in Table II for the Wine classification problem shows that both CICC-NL-NetL and CICC-NL-LSP did not perform better then standalone Neural Level (NL) results; however, they did better then the other method employed. For CICC-LSP-NetL the results outperformed both of the standalone methods. The results in Table II for the 4-bit problem shows that both CICC-NL-NetL and CICC-NL-LSP did not perform better then standalone Neural Level (NL) results; however, they did better then the other method employed. For CICC-LSP-NetL the results outperformed both of the standalone methods.

The results in Table III for the Iris problem shows that both CICC-NL-NetL and CICC-NL-LSP did not perform better then standalone Neural Level (NL) results; however, they did better then that of the other method employed. For CICC-LSP-NetL the results outperformed both of the standalone methods. The results in Table III for the Cancer problem shows that both CICC-NL-NetL and CICC-NL-LSP did not perform better then standalone Neural Level (NL) results; however, they did better then that of the other method employed. For CICC-LSP-NetL the results outperformed both of the standalone methods.

Lastly, the results in Table IV for the Heart problem shows the exact same results as the previous four problems that is, both CICC-NL-NetL and CICC-NL-LSP did not perform better then standalone Neural Level (NL) results but did better then that of the other method employed. For CICC-LSP-NetL the results outperformed both of the standalone methods.

To evaluate the scalability performance of the algorithms, we look at the number of hidden neurons used which also reflects on robustness as we are interested to know the contribution of the proposed algorithms irrespective of the neural network topology as done previously [12], [19], [31]. The CICC methods used here have shown low level scalability characteristics when compared to the standalone methods. In the Wine, Iris, Cancer and Heart problems, performances of each method tends to deteriorate when the number of hidden neurons are increased. This indicates that the nature of the problem changes when more neurons are present in the hidden layer and local search is not applicable. It also reflects bad scalability although the success rates did not change much.

In the 4-Bit problem, increasing the number of hidden neurons resulted in quite the opposite where in this case, the results were evidently dependent on the nature of the problem as it is different when compared to the real-word classification problems. Table II shows that performances of all methods on the 4-Bit problem gradually increased when the number of hidden neurons were incremented.

The competitive island framework employed here takes advantage of the best solutions of each decomposition method at a particular point in evolution time to escape from local minimum which is also credited with the good performance. In the case of using a more promising method (neural level) with methods of lesser performance (network and layer level), the results of CICC show that overall performance is not better then that of the original promising method. When using two methods of close or similar standalone performance (network and layer level), the overall performances were better then both of the standalone methods.

V. Conclusions and Future Work

Competition and collaboration between species are integral processes for survival in nature. In this paper, we implemented the competitive island cooperative co-evolution framework on feed-forward neural networks for pattern classification problems. We used two islands that employed different decomposition methods resulting in three new competitive island cooperative methods. The results show that the CICC methods showed very good performance in comparison to standalone problem decomposition methods.

The performance of the CICC method was also better in general in terms of scalability given by different number of hidden neurons of neurons. Furthermore, this enhanced performance can be credited to the nature of the CICC framework which takes advantage of the different degrees of non-separability and diversities of the decomposition methods. In contrast to conventional cooperative co-evolution, search is less likely to be trapped in a local minimum on a particular island due to collaborative features among all the islands that share best solutions.

In subsequent work, the method employed in this study can be extended by using more problem decomposition methods which can provide a more diverse solution space to compare in terms of competition and collaboration. Furthermore, it can also be applied for real-time pattern classification problems and could be extended for training other neural network architectures.

References

[1] C. G. Halpin, J. Skelhorn, and C. Rowe, "The relationship between sympatric defended species depends upon predators' discriminatory behaviour," *PloS one*, vol. 7, no. 9, p. e44895, 2012.

[2] G. Bella, *A Bug's Life: Competition Among Species Towards the Environment*, ser. Fondazione Eni Enrico Mattei Working Papers. Fondazione Eni Enrico Mattei, 2007.

[3] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evol. Comput.*, vol. 5, no. 1, pp. 1–29, Mar. 1997.

[4] M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature PPSN III*, ser. Lecture Notes in Computer Science, Y. Davidor, H.-P. Schwefel, and R. Mnner, Eds. Springer Berlin Heidelberg, 1994, vol. 866, pp. 249–257.

[5] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, no. 1, pp. 1–29, 2000.

[6] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Evolutionary Computation, Proceedings of the 2001 Congress on*, San Diego, CA, USA, Jun. 2001, pp. 1101–1108.

[7] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms," *Biosystems*, vol. 39, no. 3, pp. 263 – 278, 1996.

[8] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.

[9] L. M. Antonio and C. A. Coello Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," *2013 IEEE Congress on Evolutionary Computation*, pp. 2758–2765, 2013.

[10] N. Garcia-Pedrajas, C. Hervas-Martinez, and J. Munoz-Perez, "COVNET: a cooperative coevolutionary model for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 575–596, 2003.

[11] N. Garca-Pedrajas, C. Hervas-Martinez, and J. Munoz-Perez, "Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks)," *Neural Networks*, vol. 15, pp. 1259–1278, 2002.

[12] R. Chandra, M. Frean, and M. Zhang, "An encoding scheme for cooperative coevolutionary feedforward neural networks," in *AI 2010: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, J. Li, Ed. Springer Berlin / Heidelberg, 2010, vol. 6464, pp. 253–262.

[13] R. Chandra, "Memetic cooperative coevolution of elman recurrent neural networks," *Soft Comput.*, vol. 18, no. 8, pp. 1549–1559, 2014.

[14] F. Gomez and R. Mikkulainen, "Incremental evolution of complex general behavior," *Adapt. Behav.*, vol. 5, no. 3-4, pp. 317–342, 1997.

[15] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated neural evolution through cooperatively coevolved synapses," *J. Mach. Learn. Res.*, vol. 9, pp. 937–965, 2008.

[16] R. Chandra and M. Zhang, "Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 186, pp. 116 – 123, 2012.

[17] R. Chandra, "Competitive two-island cooperative coevolution for training Elman recurrent networks for time series prediction," in *International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, July 2014, pp. 565–572.

[18] S. Chand and R. Chandra, "Multi-objective cooperative coevolution of neural networks for time series prediction," in *International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, July 2014, pp. 190–197.

[19] R. Chandra, M. Frean, and M. Zhang, "On the issue of separability for problem decomposition in cooperative neuro-evolution," *Neurocomputing*, vol. 87, pp. 33–40, 2012.

[20] R. Chandra, M. Frean, M. Zhang, and C. W. Omlin, "Encoding subcomponents in cooperative co-evolutionary recurrent neural networks," *Neurocomputing*, vol. 74, no. 17, pp. 3223 – 3234, 2011.

[21] C. Goh, K. Tan, D. Liu, and S. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research*, vol. 202, no. 1, pp. 42 – 54, 2010.

[22] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 1, pp. 103 –127, Feb. 2009.

[23] R. Chandra, M. Frean, and M. Zhang, "Modularity adaptation in cooperative coevolution of feedforward neural networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, San Jose, CA, USA, Jul. 2011, pp. 681–688.

[24] ——, "Adapting modularity during learning in cooperative coevolutionary recurrent neural networks," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 16, no. 6, pp. 1009–1020, 2012.

[25] M. Shi, "Natural vs. unnatural decomposition in cooperative coevolution," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*. Springer, 2012, pp. 138–147.

[26] F. van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[27] Y.-j. Shi, H.-f. Teng, and Z.-q. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Advances in Natural Computation*, ser. Lecture Notes in Computer Science, L. Wang, K. Chen, and Y. S. Ong, Eds. Springer Berlin / Heidelberg, 2005, vol. 3611, pp. 1080–1088.

[28] D. Ortiz-Boyer, C. HerváMartínez, and N. García-Pedrajas, "CIXL2: a crossover operator for evolutionary algorithms based on population features," *J. Artif. Int. Res.*, vol. 24, pp. 1–48, 2005.

[29] R. Chandra, "Competition and collaboration in cooperative coevolution of elman recurrent neural networks for time series prediction," in *IEEE TRANS. On Neural Networks and Learning Systems*.

[30] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html

[31] R. Chandra, M. R. Frean, and M. Zhang, "Crossover-based local search in cooperative co-evolutionary feedforward neural networks," *Appl. Soft Comput.*, vol. 12, no. 9, pp. 2924–2932, 2012.