Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011

A Memetic Framework for Cooperative Coevolution of Recurrent Neural Networks

Rohitash Chandra, Marcus Frean and Mengjie Zhang

Abstract— **Memetic algorithms and cooperative coevolution are emerging fields in evolutionary computation which have shown to be powerful tools for real-world application problems and for training neural networks. Cooperative coevolution decomposes a problem into subcomponents that evolve independently. Memetic algorithms provides further enhancement to evolutionary algorithms with local refinement. The use of crossover-based local refinement has gained attention in memetic computing. This paper employs a cooperative coevolutionary framework that utilises the strength of local refinement via crossover. The framework is evaluated by training recurrent neural networks on grammatical inference problems. The results show that the proposed approach can achieve better performance than the standard cooperative coevolution framework.**

I. Introduction

Recurrent neural networks (RNNs) have been an important focus of research as they exhibit dynamical system properties and can be applied to difficult spatio-temporal problems [1]. Grammatical inference problems have been used to study training algorithms for recurrent neural networks [2], [3]. It has been shown that RNNs can learn and represent finite-state automata which have been verified through knowledge extraction [4].

Backpropagation-through-time (BPTT) is a standard algorithm for training recurrent neural networks [5]. Although the BPTT algorithm has been successful for a number of real world problems, it has shown limitations in learning sequences with long-term dependencies as outlined in [6]. Gradient based approaches such as backpropagation suffer from local convergence, and therefore, evolutionary computation methods have been used as an alternative for training neural networks [3]. Although evolutionary computation methods have the features of global convergence and are suitable for long-term dependency problems, they have the limitation of slower convergence.

Memetic algorithms (MAs) [7] typically combine population-based evolutionary algorithms with local refinement which provides intensification while retaining the diversification in the search process. The meme is considered to be an individual which goes through local refinement in memetic algorithms. The search for more efficient local refinement techniques has been a major focus of study in memetic computation. There is a need to use non-gradient based local search, especially in problems where gradient-based approaches fail, as in the case of training recurrent networks in problems with long-term dependencies. Crossover-based local search methods are non-gradient based and have recently gained attention [8], [9]. In crossover based local search, efficient crossover operators which have local search properties are used for local refinement with a population of a few individuals. They have shown promising results in comparison to other evolutionary approaches for problems with high dimensions [9].

Cooperative coevolution (CC) divides a large problem into smaller subcomponents and solves them independently [10]. The subcomponents are implemented as subpopulations that are genetically isolated and the only cooperation takes place in fitness evaluation. Cooperative coevolution has been shown to be effective for neuro-evolution of feedforward and recurrent networks using different problem decomposition methods that determine the way different subcomponent are composed [11].

Cooperative coevolution has the feature of decomposing a problem using several subpopulations, which provides greater diversity. Memetic computing provides further enhancement to evolutionary algorithms with local refinement. There has been much research in using local refinement with standard evolutionary algorithms. The success of local refinement in the standard evolutionary algorithm gives the motivation of using local refinement in cooperative coevolution. This paper employs the memetic cooperative coevolution framework for training recurrent neural networks. We choose the crossover based local search as the method for local refinement.

This method has been named crossover based local search in cooperative coevolution (XLCC) in [12]. XLCC has been used in training feedforward neural networks where it was shown that it outperformed canonical cooperative coevolution on classical problems [12]. In this paper, XLCC is used for training recurrent neural networks on grammatical inference problems. Crossover based local search is appropriate for local search in training recurrent neural networks as no gradient-information is needed in the search which has been a problem in learning long-term dependency problems.

The goal of this paper is to evaluate a memetic cooperative coevolutionary framework which reduces the overall training time and provides a better guarantee for convergence. The performance of XLCC is compared with standard cooperative coevolution. The main contribution of the paper is in the

Rohitash Chandra is with School of Engineering and Computer Science at Victoria University of Wellington, Wellington, New Zealand. This work was partially done when the first author was with the Department of Computing Science and Information Systems at Fiji National University, Suva, Fiji. (email: rohitash.chandra@ecs.vuw.ac.nz, rohitash_c@yahoo.com)

Marcus Frean and Mengjie Zhang are with School of Engineering and Computer Science at Victoria University of Wellington, Wellington, New Zealand. (email: {marcus.frean, mengjie.zhang}@ecs.vuw.ac.nz).

application of XLCC for training RNNs specifically for grammatical inference problems.

The Tomita grammar is used as the main grammatical inference problem. The Tomita grammar [13] has been used as a benchmark for knowledge representation and training algorithm in recurrent neural networks [2]. A specific fuzzy-finite automata problem taken from [14] is also used with the respective Tomita grammars for training the Elman style first-order RNN [15]. The G3-PCX evolutionary algorithm (Generalised generation gap with parent-centric crossover) [16] is employed in the respective CC frameworks.

The rest of the paper is organised as follows. Section II presents the background on recurrent neural networks and cooperative coevolution while Section III presents the memetic cooperative coevolution framework which employs crossover-based local search. Section IV presents experimental results and section V concludes the paper with a discussion on future work.

II. Background

A. Recurrent Neural Networks

Recurrent neural networks are dynamical systems whose next state and output depend on the present network state and input; therefore, they are particularly useful for modelling dynamical systems. Recurrent neural networks are composed of an *input layer*, a *context layer* which provides state information, a *hidden layer* and an *output layer*. Each layer contains one or more neurons which propagate information from one layer to the another by computing a non-linear function of their weighted sum of inputs.

First-order recurrent neural networks use context units to store the output of the state neurons from computation of previous time steps. The Elman architecture [15] uses the context layer which makes a copy of the hidden layer outputs in the previous time steps. The equation of the dynamics of the change of hidden state neuron activations in the context layer networks is given in Equation 1.

$$S_i(t) = g\left(\sum_{k=1}^{K} V_{ik}S_k(t-1) + \sum_{j=1}^{J} W_{ij}I_j(t-1)\right) \qquad (1)$$

where $S_k(t)$ and $I_j(t)$ represent the output of the context state and input neurons, respectively and V_ik and W_ij represent their corresponding weights. $g(.)$ is a sigmoid discriminant function.

Grammatical inference problems have been used to study training algorithms for knowledge representation in recurrent neural networks. There is no feature extraction necessary in order for recurrent neural networks to learn these languages. It has been demonstrated through knowledge extraction that RNNs can represent finite-state automata [4]. Deterministic finite-state automata (DFA) and fuzzy finite-state automata (FFA) are the major grammatical inference problems studied in the past [4]. The Tomita grammar [13] is an example of a DFA and has been used as a benchmark problem in order to evaluate RNN training algorithms and architectures [2]. The Tomita grammar consists of seven regular languages. An illustration of two selected DFA's from the Tomita grammar is shown in Figure 1.

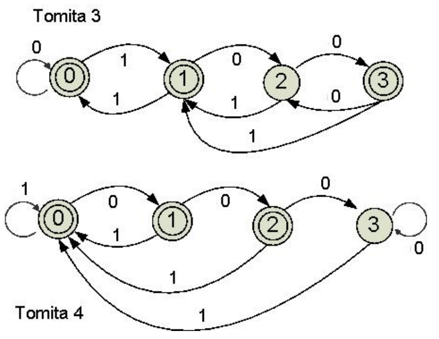


Fig. 1. Deterministic Finite-State Automata from the Tomita grammar: Double circles in the figure show accepting states while rejecting states are shown by single circles. State 1 is the automatons start state.

Figure 2 shows an example of a FFA with its corresponding deterministic acceptor which is used for training recurrent neural networks. This FFA has been used by [14] to show that RNNs can be trained by evolutionary algorithms.

B. Cooperative Coevolution

In the evolutionary process of nature, different species compete in order to survive with the given resources. The individuals of a particular group of species mate amongst themselves in order to produce stronger individuals. However, mating in between different species is not feasible. The cooperative coevolution framework is nature inspired where species are represented as subcomponents that are implemented as subpopulations [17].

The subpopulations in the cooperative coevolution framework are evolved separately and the cooperation only takes place for fitness evaluation for the respective individuals in each subpopulation. The size of a subcomponent and the way the problem is decomposed into subcomponents is dependent on the problem. Problem decompoistion has been a major focus of research in cooperative coevolution [11], [18]. The way the algorithm cooperatively evaluates each subcomponent is known as fitness estimation. A method for estimating fitness has been proposed by Potter and Jong [17]. This method obtains the fitness of each individual in a subpopulation by combining it with the best individuals for the rest of the subpopulations.

C. Problem decomposition in cooperative coevolution

Problem decomposition in cooperative coevolution determines the way a subcomponent is composed which is dependent on the neural network architecture. We also refer to problem decomposition as encoding scheme. The major encoding schemes include those on the *synapse level* and the *neuron level*. In the synapse level, a single subpopulation is used to represent each weight or synapse in the network as shown in [11]. The neuron level designates each neuron in the hidden layer as the main reference point for the respective

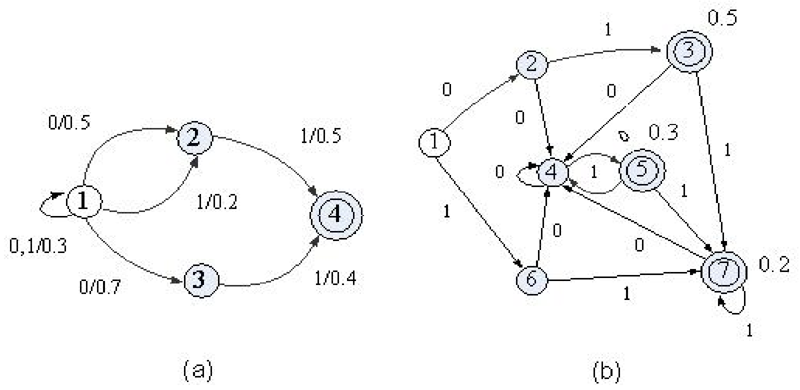


Fig. 2. The fuzzy finite-state automata (a) and its equivalent deterministic acceptor shown in (b). The accepting states are labelled with a degree of membership. State 1 is the automatas start state; accepting states are drawn with double circles [14].

subcomponent. Each subcomponent consists of the incoming and outgoing connections. An example of this method is *enforced subpopulations* (ESP) which has been used for training recurrent neural networks [19]. The neuron based subpopulation (NSP) [18], [20] decomposes the network to the neuron level and unlike ESP, the subcomponents do not include the outgoing weight links associated with a neuron as shown in Figure 3.

In this paper, the NSP [20] is used for training RNNs where each subpopulation in a layer is composed of the following.

1) Hidden layer subpopulations: weight-links from each neuron in the $hidden_j$(t) layer are connected to all $input_i$(t) neurons and the bias of $hidden_j$(t)
2) Context layer (Recurrent) subpopulations: weight-links from each neuron in the $hidden_j$ (t) layer are connected to all hidden neurons in previous time step $hidden_j$(t-1)
3) Output layer subpopulations: weight-links from each neuron in the $output_k$(t) layer are connected to all $hidden_j$(t) neurons and the bias of $output_k$(t)

where $input_i$(t) is the input neuron i at time t, $hidden_j$(t) is the hidden neuron j at time t and $output_k$(t) is the output neuron k at time t. Each neuron in the hidden and output layer acts as a reference point to its subpopulations. Note that the NSP can be easily extended for multiple hidden layers.

III. Memetic Cooperative Coevolution Framework

The term "memetic algorithm" was introduced by Moscato in his report [21] where he viewed memetic algorithms as a population based hybrid genetic algorithm with local refinement. The local refinement or individual learning procedure is similar to the concept of the *meme* that undergoes social or cultural evolution. In the rest of the discussion, we will refer to the meme as an individual which goes through local refinement. Similarly, we will refer to individual learning as local search.

In the previous sections, we have discussed how the neural network learning problem can be decomposed into subcomponents which are implemented as subpopulations that coevolve separately and cooperate only for fitness evaluations. It is important to note that training neural networks through any evolutionary computation framework is computationally expensive in terms of function evaluations. The function evaluation depends on the type and size of the network architecture and the size of the training data. Moreover, the number of features in the training data also adds to the computational cost. Therefore, it is important to investigate on ways by which the number of function evaluations can be reduced while achieving a global solution.

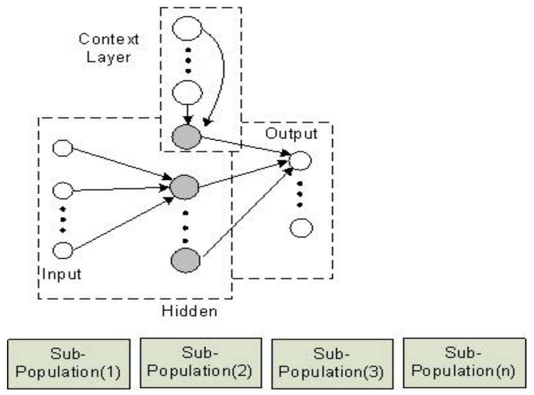


Fig. 3. The NSP encoding scheme. The same encoding is used in the rest of the neurons in the hidden and output layer [20] . Note that the NSP can be easily extended for multiple hidden layers.

Memetic based approaches have mainly been developed using evolutionary algorithms which have a population of individuals. In the case of building a memetic computation framework for multiple subpopulations in cooperative coevolution, we need to consider computational costs of having individual refinement for each subpopulation. The respective individuals in a subpopulation that undergo individual refinement only represents a subset of the large problem. In order to apply local refinement, the respective individual has to be concatenated with the best individuals in the rest of the subpopulations. Therefore, given n subpopulations, n local refinement would be required which would add to computational cost.

We propose a framework which efficiently takes in advantage of the local search in consideration of the computational cost of having a separate local search for every subpopulation. Rather than employing a local search for each subpopulation, our proposed framework employs local search only when a *cycle* is complete. The completion of a cycle in cooperate coevolution indicates that all the respective subpopulations have been evolved for a fixed number of generations in a round-robin fashion.

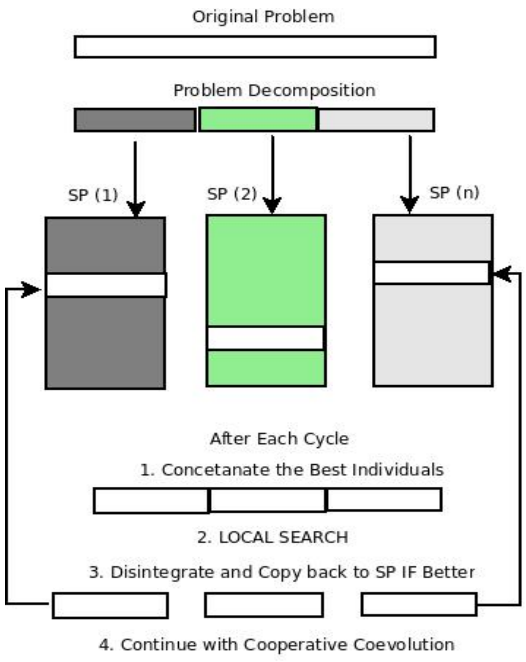

Original Problem
Problem Decomposition
SP (1)
SP (2)
SP (n)
After Each Cycle
1. Concetanate the Best Individuals
2. LOCAL SEARCH
3. Disintegrate and Copy back to SP IF Better
4. Continue with Cooperative Coevolution

Fig. 4. The XLCC framework for evolving recurrent networks. The LOCAL SEARCH is employed after concatenating the best individuals from each subpopulation (SP) at the end of each cycle.

The details of the memetic cooperative coevolution framework for evolving recurrent networks is given in Algorithm 1. The algorithm assumes that it has been given the best parameters for the evolutionary algorithm such as its population size, crossover and mutation rate.

The algorithm begins by encoding the neural network into the subpopulation according to the respective cooperative coevolution encoding scheme (either ESP, CoSyNE or NSP [11], [20]). The specific encoding scheme for this work is NSP which has already been discussed earlier. All the individuals of the respective subpopulations are seeded with random real values in a range. Each individual is then concatenated with the best individuals from the rest of the subpopulations and then decoded into a neural network and evaluated as done in [17]. The fitness function is defined by the *mean-squared-error* of the recurrent network. The algorithm proceeds as a standard evolutionary algorithm which employs genetic operators such as selection, crossover and mutation to create new offspring for all the subpopulations. After the completion of each cycle, the best individuals from all the subpopulations are concatenated into a meme which is further refined as shown in Figure 4. The meme is then refined using crossover based local search where a population of few individuals are evolved for a given number of generations. The duration of local refinement is known as the *Local Search Intensity*. The refined meme is then disintegrated and copied to the respective subpopulations. The refined meme replaces the worst individuals in each of the subpopulations. Note that even if the refinement meme is not improved, it replaces the worst individual, as it may have features which will be used later in evolution using genetic operators. Although crossover based local search is used as the designated algorithm for local refinement, the framework can employ any other local search method.

A. G3-PCX genetic algorithm for local search

The G3-PCX differs from a standard genetic algorithm in terms of selection as it employs the generalised generation gap model for selection [16]. In G3-PCX, the whole population is initially randomly seeded and evaluated similarly to the standard genetic algorithm. The difference lies in the evolution phase where a small subpopulation is made of few chosen parents and children. At each generation, only the subpopulation is evaluated rather than the whole population. The children with their new fitness become part of the bigger population. The best individual in the population is retained at each generation. The parent centric crossover operator is used in creating an offspring based on orthogonal distance between the parents. The parents are made of female and male component. The *female* parent points to search areas and *male* parent is used to determine the extent of search of the areas pointed by the female. The genes of the offspring extract values from intervals associated in the neighbourhood of the female and male using probability distribution. The range of this probability distribution depends on the distance among the genes of the male and the female parent. The parent-centric crossover operator assigns more probability to create an offspring near the female than anywhere else in the space. The major strength of the parent-centric crossover operator is that it behaves like a mutation operator and at the same time retains diversity and is self-adaptive.

IV. Simulation and Analysis

This section presents an experimental study on the XLCC for evolving recurrent network for grammatical inference problems. The G3-PCX evolutionary algorithm [16] is employed in the respective subpopulations and also used for crossover based local search. The crossover based local search has a population of 20 individuals. The cooperative coevolutionary framework has 100 individuals in all subpopulations. These parameters were determined in trial experiments.

The G3-PCX algorithm employs the mating pool size of 2 offspring and the family size of 2 parents. This setup has been used in [16] and has shown good results for general optimisation problems. The subpopulations are seeded with

Alg. 1 The Memetic Cooperative Coevolution Framework

```
– Encode the neural network using an appropriate problem decomposition method
– Randomly initialise all subpopulations
– Cooperatively evaluate all individuals in each subpopulation

while NOT termination do
  for each subpopulation do
        i) Create new individuals using genetic operators
        ii) Update subpopulation
  end for

        – Concatenate the best individuals from each subpopulation into meme $M$
        – Local refinement on $M$ according to the LSI

  i) Decompose the refined individuals for respective subpopulation
  ii) Replace the worst individuals of the respective subpopulations with the decomposed individual
end while
```

random real numbers in the range of [-5, 5] in all experiments. The cooperative coevolution framework employs NSP for problem decomposition as shown in Figure 3.

The FFA shown in Figure 2 taken from [14] is used. We generate a training dataset by presenting strings of increasing lengths of 1 to 7 to the deterministic acceptor in Figure 2 (b) and record the corresponding output for each sample. Note that for every string length, all the possible strings are generated. Thus, the generated training set consists of 255 strings. Similarly, we generate the testing dataset with string lengths of 8 to 14 using the same FFA. The recurrent neural network topology for the FFA is as follows: 1) one neuron in the input layer, 2) two output neurons in the output layer representing the 4 fuzzy output states of the FFA. The FFA problem uses 5 neurons in the hidden layer of the RNN.

The RNN is trained until 100 percent of the training samples are correctly classified or when the maximum number of function evaluation is reached. This value was determined in trial experiments. For the FFA problem, the maximum number of function evaluations is pre-set to 5000.

Similarly, we generated the training and testing dataset from the Tomita language given in Figure 1. We used Tomita 2 (T2) , Tomita 3 (T3) , and Tomita 4 (T4). In this case, the training and testing data is generated by presenting random strings of length 10 to 15 for each Tomita language. The training and testing dataset contains 250 (125 positive and 125 negative) string samples. The RNN uses 2 neurons in the hidden layer for T2 and 3 neurons for the hidden layer for T3 and T4. The number of hidden neurons is fixed for all problems. The maximum number of function evaluations in T2 is 2000, and T3 and T4 is 5000.

We report the training behaviour of the respective algorithms in terms of number of function evaluations and the number of successful runs which indicates the efficiency of the respective methods. The goal of each algorithm is to obtain the maximum success rate out of 100 experiments with the least optimisation time.

A. Local Search Intensity and Frequency

The two main parameters in the XLCC are the *Local Search Intensity (LSI)* and *Local Search Frequency (LSF)*. The LSI determines how long the local refinement is employed and the frequency determines when to apply local refinement, i.e, after how many consecutive cycles of undergoing cooperative coevolution.

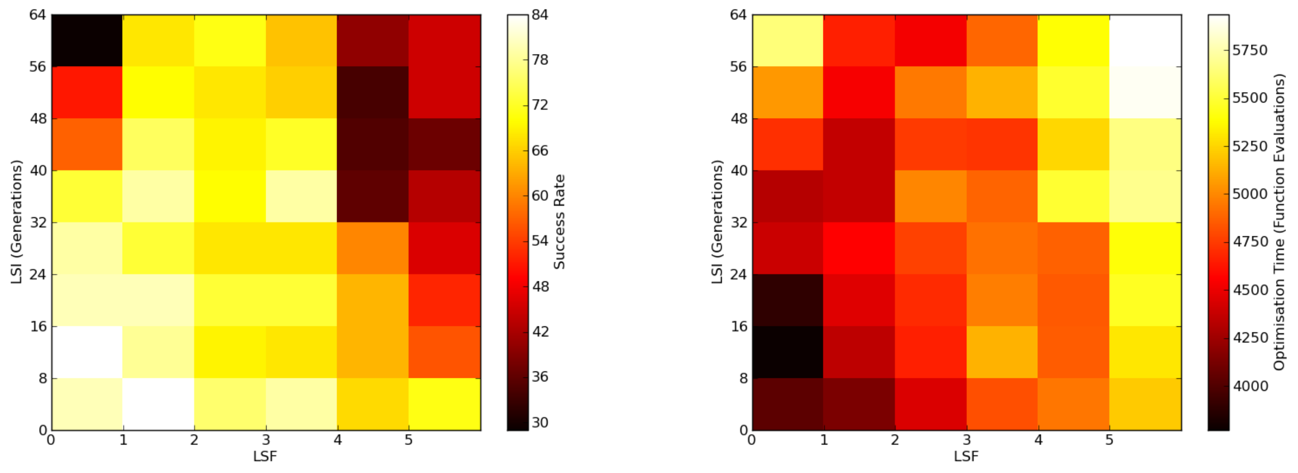
It is important to study the behaviour of XLCC and explore the relationship between the LSI and LSF. The results in Figure 5(a) and 5(b) show the heat-maps which report the behaviour of XLCC given different combinations of LSI and LSF for the T3 problem. The LSI is given by the number of generations employed in the crossover-based local search.

Figure 5(a) shows that the LSF of 1 gives high success rates particularly for LSI of 8 – 32 generations. The optimal is at 16 generations. Good performance in terms of function evaluations for the LSI of 16 generations is shown in Figure 5(b). In general, the figure shows that as the LSF is increased, the optimisation time increases with slight differences in the success rate.

Figures 6 and 7 give further details to the results in Figure 5 for the T3 problem. The figures also give the results for the FFA, T2, T4 and T5 grammatical inference problems.

Figure 6 evaluates the behaviour of XLCC on different LSF for the three problems. The LSI of 8 generations is used. The LSF of 1 gives the best results in terms of least optimisation time taken as shown in Figure 6(a) with high success rates in Figure 6(b). The LSF higher than 1 requires more optimisation time.

Figure 7 evaluates the behaviour of XLCC on the fixed LSF of 1 with different values for LSI. Here, we first look at the success rate and choose the corresponding LSI if the optimisation time is acceptable. The FFA problem shows best performance in terms of the success rate in Figure 7(a)



(a) Success Rate in the T3 Problem

(b) Optimisation time in the T3 Problem

Fig. 5. The heatmap shows the performance of XLCC on different Local Search Frequency (LSF) and Local Search Intensity (LSI) for the T3 problem. The number of successful runs is evaluated in (a) while the optimisation time in terms of the number of function evaluations is evaluated in (b). The goal of XLCC is to obtain maximum success with the minimum number of function evaluations. The LSF of 1 and LSI of 16 shows the best success rate in (a) with corresponding least number of function evaluations in (b).

and good optimisation time in Figure 7(b) for the LSI of 40 generations. The T2 problem shows best performance with LSI of 64 generations. T3 has good performance with LSI of 16 generations. In T3, the success rate gets lower as the LSI increases beyond 16 generations. In T4, the LSI of 32 and 64 have the same success rate, however, LSI of 64 obtains better performance in terms of the optimisation time. Therefore, the LSI of 64 is chosen.

Table I shows a comparison of XLCC with standard cooperative coevolution. This table consists of the best results in terms of the highest success rate and least optimisation time from Figure 7. Note that the NSP problem decomposition method with G3-PCX is used in both methods. The results show that performance of XLCC has improved in almost all the problems in terms of the optimisation time and the success rate.

TABLE I
COMPARISON OF XLCC WITH CC

Problem	Method	LSI	LSF	Time	Suc. Rate
FFA	CC	–	–	4243_{597}	92
	XLCC	40	1	3319_{421}	75
T2	CC	–	–	1836_{34}	80
	XLCC	64	1	468_{44}	100
T3	CC	–	–	3560_{467}	72
	XLCC	32	1	2693_{321}	92
T4	CC	–	–	3292_{231}	90
	XLCC	64	1	643_{56}	100

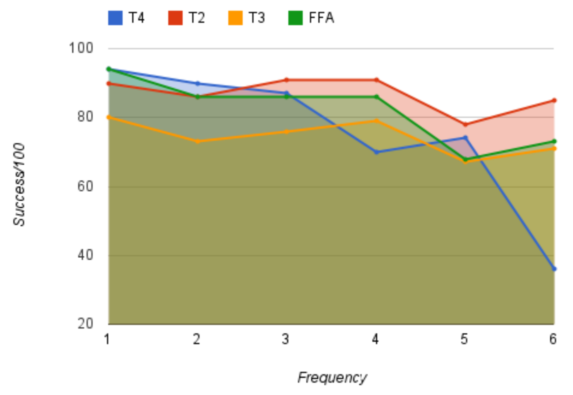
B. Discussion

The results in general show that the local search frequency of 1 gives the best performance in all four problems which implies that the local search has to be applied most frequently. The memetic cooperative coevolution framework has to take maximum advantage of local refinement after every cycle in cooperative coevolution in order to balance the global and local search.

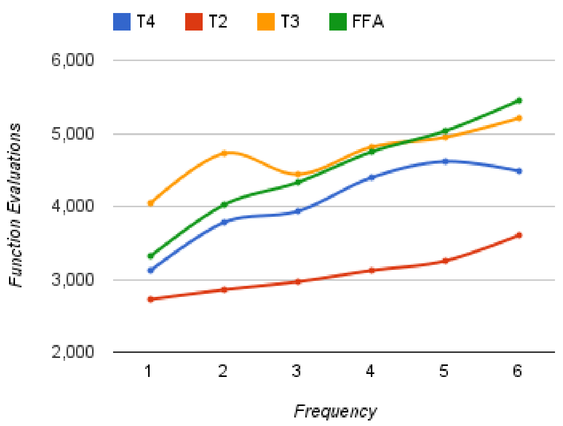
The results also show that the local search intensity is an important parameter of XLCC and its depth on search is dependant on the nature of the problem.

The trend is similar in FFA and T3 problems where for a certain interval, good performance is achieved and later with deeper local search, poorer performance is reported. In T4, LSI of 64 gives the best results. These similarities are due to the nature of the problems. T2 and T4 required deeper local search than FFA and T3 as they seem to be problems where more global search using cooperative coevolution is needed.

In general, comparison of the XLCC framework with standard CC framework in Table I shows improved performance in all given problems. The success rate and the total number of function evaluations is improved. Only for the FFA problem, the success rate is smaller, however, the number of function evaluations has improved. This implies that it is important to employ local search in cooperative coevolutionary framework for training recurrent neural networks. The results have clearly shown that the depth of search is an important parameter, and must be tailored for each problem prior to the deployment of the memetic cooperative coevolution framework. In more difficult problems (FFA and T3), a shallow local search is needed while in easier problems (T2, and T4), a deeper local search greatly improves the performance.

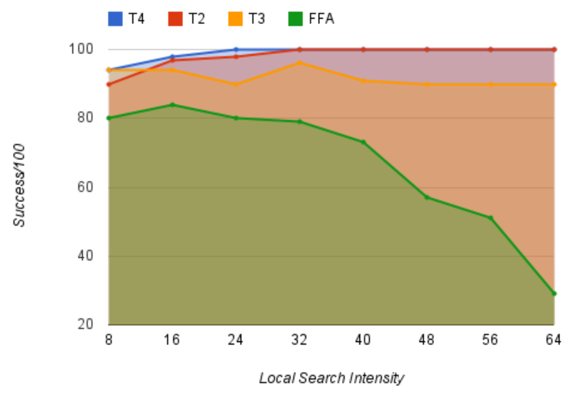


(a) Success Rate for evaluating the LSF

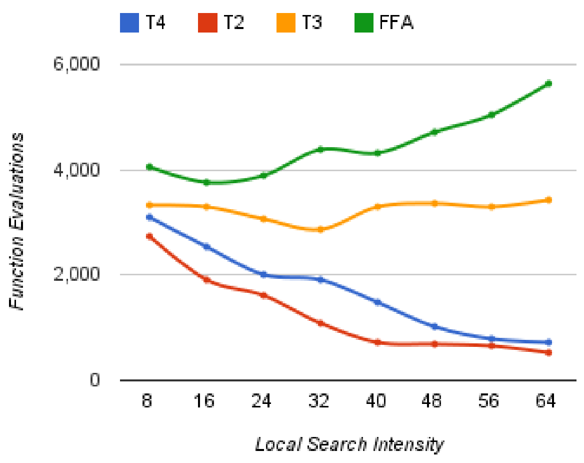


(b) Optimisation time for evaluating the LSF

Fig. 6. The evaluation of the Local Search Frequency (LSF) for the FFA, T2, T3 and T4 grammatical inference problems. The LSI of 8 generations is used as a fixed parameter in all problems. Note that the LSF of 1 shows the highest success rate and least number of function evaluations for all problems.



(a) Success Rate for evaluating the LSI



(b) Optimisation time for evaluating the LSI

Fig. 7. The evaluation of the LSI for the FFA, T2, T3 and T4 grammatical inference problems. Note that the LSF of 1 is used in all the problems.

V. Conclusions and Future Work

This paper applied XLCC to recurrent neural networks. The framework used memetic computation which mimics evolutionary biology where memes undergo lifetime learning or social evolution. In the XLCC framework, the local refinement done by crossover-based local search can be seen as lifetime learning which enhances the evolutionary process by producing stronger individuals when applied in the appropriate search process.

The main problem in this study was to efficiently utilise local refinement in the respective subpopulations. This problem has been efficiently solved through the framework which decomposes the locally refined solution and incorporates it into the subpopulations. The memetic framework progresses with a global search through evolution of standard cooperative coevolution and as a specified time is reached (in terms of local search frequency), the algorithm adapts itself by incorporating local refinement in the subpopulations. Even though the best individual from the crossover-based local search has not shown major improvement, it is always added to the cooperative coevolutionary subpopulations.

The XLCC framework has performed better for all the given problems when compared to the performance of standard cooperative coevolution. This opens the road for further research in using other local refinement methods. Back-propagation can also be utilised. It can replace or be added with the crossover-based local search for local refinement. This will enable the memetic cooperative coevolution framework to incorporate gradient information into the evolutionary search process.

The major limitation of the XLCC framework is the computational cost required in parameter setting; i.e. optimal values for the frequency and the local search intensity. In the case of grammatical inference problems, the frequency of 1 has given the optimal solution in all the problems, however, the local search intensity varied from 16 to 64 generation in different problems. In similar problems, the local search intensity has to be evaluated beforehand. A heuristic can be proposed where the minimum and maximum generation of local search intensity can be fixed, and the search terminates when there is no improvement for a fixed number of generations.

Recurrent neural networks have faced problems in learning long term dependencies with BPTT. For this reason, BPTT has not been used in the XLCC framework, however, its usage is possible according to the application problem at hand. The given XLCC RNN approach is suitable for any long-term dependency problem where BPTT fails.

The XLCC approach is general, therefore, any evolutionary algorithm can be used with a suitable local search. In future work, it will be interesting to apply the XLCC for real-world applications such as speech recognition using recurrent neural networks. The same paradigm can also be used to train other recurrent network architectures and also be extended for general global optimisation problems.

References

[1] C. L. Giles, C. Omlin, and K. K. Thornber, "Equivalence in knowledge representation: Automata, recurrent neural networks, and dynamical fuzzy systems," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1623–1640, 1999.

[2] M. A. Castao, E. Vidal, and F. Casacuberta, "Finite state automata and connectionist machines: A survey." in *IWANN*, ser. Lecture Notes in Computer Science, J. Mira and F. S. Hernndez, Eds. Springer, 1995, pp. 433–440.

[3] P. Angeline, G. Saunders, and J. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *Neural Networks, IEEE Transactions on*, vol. 5, no. 1, pp. 54 –65, jan 1994.

[4] K. K. T. C. W. Omlin and C. L. Giles, "Fuzzy finite state automata can be deterministically encoded into recurrent neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 76–89, 1998.

[5] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[6] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[7] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Tech. Rep., 1989.

[8] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing." *Evol. Comput.*, vol. 12, no. 3, pp. 273–302, 2004.

[9] D. Molina, M. Lozano, C. Garca-Martnez, and F. Herrera, "Memetic algorithms for continuous optimisation based on local search chains," *Evol. Comput.*, vol. 18, no. 1, pp. 27–63, 2010.

[10] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1994, pp. 249–257.

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

[12] R. Chandra, M. Frean, and M. Zhang, "A memetic framework for cooperative co-evolutionary feedforward neural networks," School of Computing and Engineering, Victoria University of Wellington, Wellington, New Zealand, Technical Report ECSTR10-22, 2010.

[13] M. Tomita, "Dynamic construction of finite automata from examples using hill-climbing," in *Proceedings of the Fourth Annual Cognitive Science Conference*, Ann Arbor, MI., 1982, pp. 105–108.

[14] A. Blanco, M. Delgado, and M. C. Pegalajar, "A real-coded genetic algorithm for training recurrent neural networks," *Neural Netw.*, vol. 14, no. 1, pp. 93–105, 2001.

[15] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[16] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evol. Comput.*, vol. 10, no. 4, pp. 371–395, 2002.

[17] 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.

[18] 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, 2011, vol. 6464, pp. 253–262.

[19] F. J. Gomez, "Robust non-linear control through neuroevolution," PhD Thesis, Department of Computer Science, The University of Texas at Austin, Technical Report AI-TR-03-303, 2003.

[20] R. Chandra, M. Frean, M. Zhang, and C. Omlin, "Building subcomponents in the cooperative coevolution framework for training recurrent neural networks," School of Engineering and Computer Science, Victoria University of Wellington, New Zealand, Technical Report ECSTR09-14, 2009.

[21] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Caltech Concurrent Computation Program, Technical Report 826, 1989.