Soft Comput
DOI 10.1007/s00500-013-1160-1

METHODOLOGIES AND APPLICATION

Memetic cooperative coevolution of Elman recurrent neural networks

Rohitash Chandra

Abstract Cooperative coevolution decomposes an optimisation problem into subcomponents and collectively solves them using evolutionary algorithms. Memetic algorithms provides enhancement to evolutionary algorithms with local search. Recently, the incorporation of local search into a memetic cooperative coevolution method has shown to be efficient for training feedforward networks on pattern classification problems. This paper applies the memetic cooperative coevolution method for training recurrent neural networks on grammatical inference problems. The results show that the proposed method achieves better performance in terms of optimisation time and robustness.

Keywords Recurrent neural networks · Memetic algorithms · Local search · Cooperative coevolution · Grammatical inference

1 Introduction

Recurrent neural networks are dynamical systems that have been successful in problems that include time series prediction, classification, language learning and control (Robinson 1994; Seyab and Cao 2008). Finite-state machines have been used to demonstrate knowledge representation and learning in recurrent networks (Giles et al. 1995).

Cooperative coevolution (CC) divides a problem into subcomponents (Potter and Jong 1994) that are represented using sub-populations which are genetically isolated. Cooperative coevolution has shown to be effective for neuro-evolution of feedforward and recurrent networks (Gomez 2003; Gomez et al. 2008; Chandra et al. 2011c). Cooperative coevolution has the feature of decomposing a problem using several subpopulations which provides greater diversity and increased global search features (Chandra et al. 2012c).

Problem decomposition is a major issue in the use of cooperative coevolution for neuro-evolution. It is essential to break the network into subcomponents that have the least interactions amongst themselves (Chandra et al. 2012c). There are two major problem decomposition methods for neuro-evolution that decomposes the network on the *neuron* and *synapse level*. In synapse level problem decomposition, the neural network is decomposed to its lowest level where each weight connection (synapse) forms a subcomponent. Examples include cooperatively co-evolved synapses neuro-evolution (Gomez et al. 2008) and neural fuzzy network with cultural cooperative particle swarm optimisation (Lin et al. 2009). In neural level problem decomposition, the neurons in the network act as the reference point for the decomposition. Examples include enforced subpopulations (Gomez and Mikkulainen 1997; Gomez 2003) and neuron-based subpopulation (Chandra et al. 2010, 2011c). Adaption of problem decomposition during neuro-evolution has shown promising results for feedforward and recurrent neural networks (Chandra et al. 2011b, 2012a). Adaptation strategies can ensure different levels of diversification and intensification at different stages of the evolutionary search (Chandra et al. 2012a,c).

Memetic algorithms (MAs) combine population-based evolutionary algorithms with local search methods that are also known as individual learning or local refinement (Moscato 1989a). The search for efficient and robust local refinement procedures has been the focus of memetic algorithms (Smith 2007; Molina et al. 2010). Memetic algorithms

Communicated by G. Acampora.

R. Chandra (✉)
Faculty of Science Technology and Environment, School of Computing, Information and Mathematical Sciences, University of the South Pacific, Suva, Fiji
e-mail: c.rohitash@gmail.com

Published online: 24 October 2013

have been used for solving optimization problems with computationally expensive fitness functions (Zhou et al. 2007), large scale combinatorial optimization problems (Tang et al. 2007), enhancing e-learning environments (Acampora et al. 2011b) and ontology alignment problem (Acampora et al. 2012). Other applications include combinatorial optimisation problems, machine learning such as training neural networks, molecular optimisation problems, electronics and engineering, and other optimisation problems as discussed in Moscato (2003).

Crossover-based local search has shown good performance in memetic algorithms (Molina et al. 2010). A study on the balance of diversification using cooperative coevolution and intensification using local search has been successful for cooperative coevolution of feedforward networks on pattern classification problems (Chandra et al. 2012b). It is important to investigate how often to apply local search *(local search interval)* and for how long to apply them *(local search intensity)*. Our recent work presented a memetic cooperative coevolution method for training feedforward networks has been called crossover-based local search in cooperative coevolution (XLCC) Chandra et al. (2012b). It would be interesting to study the performance of XLCC for training recurrent neural networks since it has a different search landscape as feedback connections are present.

This paper applies XLCC for training Elman recurrent networks (Elman 1990) on a set of grammatical inference learning problems to evaluate the training time and guarantee for convergence in terms of scalability and robustness. It extends the results published in Chandra et al. (2011a) by using a heuristic to determine the local search intensity during the evolutionary process and testing the method on different number of hidden neurons reflecting on scalability and robustness.

The rest of the paper is organised as follows. Section 2 gives a background on memetic algorithms, cooperative coevolution and recurrent networks. Section 3 presents the memetic cooperative coevolution framework that features crossover-based local search. Section 4 presents experimental results and Sect. 5 concludes the paper with a discussion on future work.

2 Background

2.1 Memetic algorithms

Global search traverses over several neighbourhood of solutions while local search limits itself within a single solution neighbourhood. The neighbourhood $N(v)$ of a vertex v is the sub-graph that consists of the vertices adjacent to v (not including v itself) (Watts 1999).

Local search is also viewed as hill climbing that refines the solution. Evolutionary search methods begin with global search that contains large difference between candidate solutions in the population. As the search progresses, with evolutionary operators such as selection and recombination, the search points to a single solution neighbourhood and the candidate solutions are closer to each other. Local search is encouraged towards the end of the search when the distance between the candidate solutions get smaller. The same recombination operators used in the beginning of the search may not be applicable during the end, and therefore, adaptation is important. This is the main reason adaptation of the recombination operators and local search methods play an important role during the evolutionary process.

Meta-heuristics refer to the family of search algorithms that have extended basic heuristic methods by extending exploration capabilities (Glover and Kochenberger 2003). Memetic algorithms use master meta-heuristics for diversification and a subordinate meta-heuristic for intensification. Memetic algorithms address the shortcomings of evolutionary algorithms in balancing diversification and intensification (Moscato 1989a). Memetic algorithms also include the combination of evolutionary algorithms with problem dependent heuristics and approximate methods and special recombination operators (Moscato 2003). Memetic algorithms are often referred to as Baldwinian evolutionary algorithms, Lamarckian evolutionary algorithms, cultural algorithms or genetic local search.

Memetic algorithms have typically used evolutionary algorithms for diversification combined and local search methods such as hill-climbing for intensification. Initial work was done by Moscato who used a genetic algorithm for diversification with local search for intensification (Moscato 1989b). Lozano et al. (2004a) presented memetic algorithm with crossover hill-climbing as a local search. The crossover operator repeatedly produces a fixed number of offspring from which the best is selected.

Ong and Keane (2004) presented a meta-Larmarckian memetic framework where several different types of local search algorithms are employed during evolution. Initially, all local search algorithms are given a chance and hence their fitness is measured which is kept in future so that roulette wheel selection can be used to select the local search. The method showed high quality and efficient performance on classic benchmark functions for continuous optimisation and a real world aerodynamic design problem. Smith (2007) presented a review on co-evolving memetic algorithms in which a rule-based representation of local search is coadapted alongside candidate solutions within a hybrid evolutionary algorithm. Nguyen et al. (2009) presented a probabilistic memetic framework that analyses the probability of the process of individual learning in locating global optimum. Agent-based machine learning methods has been used

to address adaptation in memetic algorithms (Acampora et al. 2011a).

It has been found that crossover based local search (Lozano et al. 2004b; Molina et al. 2010), gave good performance for real parameter optimisation problems. In crossover based local search, efficient crossover operators that have local search properties are used with a population of a few individuals. They have shown promising results in comparison with other evolutionary methods for optimisation problems with high dimensions (Molina et al. 2010).

2.2 Cooperative coevolution

Cooperative coevolution (CC) is an evolutionary computation method inspired from nature which divides a large problem into subcomponents and solves them collectively in-order to solve the large problem (Potter and Jong 1994).

The original cooperative coevolution algorithm (Potter and Jong 1994) can be summarised as follows.

1. *Problem decomposition:* Decompose a high dimensional problem into subcomponents that can be solved by conventional evolutionary algorithms. The subcomponents can vary in sizes and are often expressed as subpopulations.
2. *Subcomponent optimisation:* Evolve each subcomponent separately by an evolutionary algorithm where evolutionary operators such as crossover and mutation are restricted to a subcomponent and do not affect other subcomponents.
3. *Fitness evaluation:* Fitness of individuals in each of the subcomponents are evaluated cooperatively with representative examples from the other subcomponents.

There are two major problem decomposition methods for cooperative coevolution of recurrent neural networks. In Synapse level problem decomposition, the network is decomposed to its lowest level where each weight link (synapse) in the network forms a subcomponent. The number of subcomponents depend on the number of weights and biases (Gomez et al. 2008; Lin et al. 2009).

In Neural level problem decomposition, each neuron in the hidden layer is used as a major reference point for each subcomponent. Therefore, the number of hidden neurons determines the number of subcomponents. Neural level decomposition has been efficient for training recurrent neural networks for grammatical inference problems (Chandra et al. 2011c). Each subcomponent consists of weight links associated with a neuron in the hidden, state (recurrent), and output layer as shown in Fig. 1. Therefore, each subcomponent is implemented as a sub-population and defined as follows:

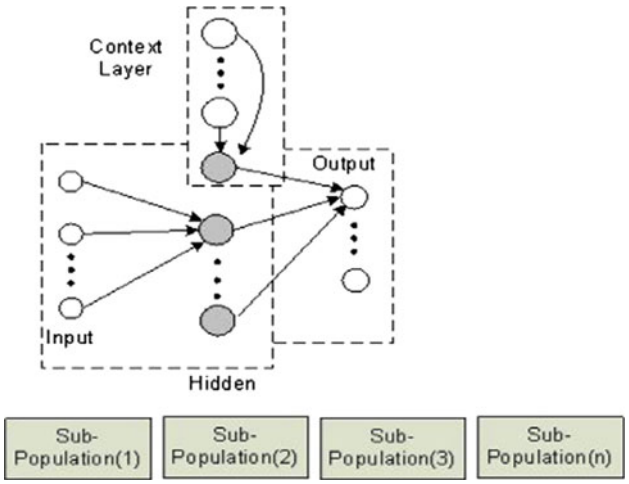


Fig. 1 Each neuron in the hidden and output layer acts as a reference point for each subcomponent (Chandra et al. 2011c)

1. Hidden layer sub-populations: weight-links from each neuron in the $hidden(t)$ layer connected to all $input(t)$ neurons and the bias of $hidden(t)$, where t is time.
2. State (recurrent) neuron sub-populations: weight-links from each neuron in the $hidden(t)$ layer connected to all hidden neurons in previous time step $hidden(t-1)$.
3. Output layer sub-populations: weight-links from each neuron in the $output(t)$ layer connected to all $hidden(t)$ neurons and the bias of $output(t)$.

2.3 Recurrent neural networks

Recurrent neural networks have been an important focus of research as they can be applied to difficult problems involving time-varying patterns. They are suitable for modelling temporal sequences. A detailed study on the theoretical foundations, design and application of recurrent neural networks is done in Haykin et al. (2006), Kolen and Kremer (2001), Medsker and Jain (1999).

First-order recurrent neural networks use context units to store the output of the state neurons from computation of the previous time steps. The context layer is used for computation of present states as they contain information about the previous states. Manolios and Fanelli have shown that first-order recurrent networks can learn and represent deterministic finite-state automata (Manolios and Fanelli 1994). The Elman recurrent network architecture have been trained using evolutionary algorithms (Pham and Karaboga 1999). The computational power of Elman recurrent networks has been studied and it has been shown that their dynamical properties can represent any finite-state machine (Kremer 1995).

The Elman architecture (Elman 1990) employs a context layer which makes a copy of the hidden layer outputs in the previous time steps. The dynamics of the change of hidden

state neurons activation in Elman style recurrent networks is given by Eq. (1).

$$y_i(t) = f\left(\sum_{k=1}^{K} v_{ik}\ y_k(t-1) + \sum_{j=1}^{J} w_{ij}\ x_j(t-1)\right) \quad (1)$$

where $y_k(t)$ and $x_j(t)$ represent the output of the context state neuron and input neurons, respectively. v_{ik} and w_{ij} represent their corresponding weights. $f(.)$ is a sigmoid transfer function.

3 Memetic cooperative coevolution framework for recurrent networks

Memetic algorithms have been mainly developed using evolutionary algorithms that have a single population of individuals. In the case of building a memetic computation framework for several sub-populations in cooperative coevolution, we need to consider computational costs of having local search for each sub-population. In order to apply local search, the respective individual has to be concatenated with the best individuals in the rest of the sub-populations. Therefore, given n sub-populations, n local searches are required which adds to the computational cost as shown in Fig. 2.

Our previous work presented a memetic framework that takes advantage of the local search while considering the computational cost of having a separate local search for every sub-population (Chandra et al. 2012b). It employs local search only when all the sub-populations in cooperative coevolution have been evolved. The two main parameters of the memetic framework are the local search intensity (LSI) and local search interval (LS-Interval). The LSI determines how long the local refinement is done and the interval determines when to apply local refinement, i.e, after how many consecutive cycles of undergoing standard cooperative coevolution. For instance, the LS-Interval of 3 means that local refinement will be employed once with given LSI after every 3 cycles.

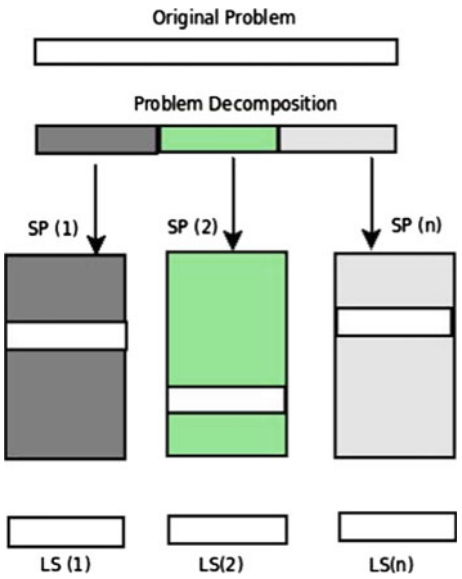


Fig. 2 Problem faced by cooperative coevolution in employing n local searches (LS) to each sub-population (SP)

Alg. 1 Memetic Cooperative Coevolution Framework

– Encode the neural network using an appropriate encoding scheme
– Randomly initialise all sub-populations
– Cooperatively evaluate each sub-population

while NOT termination **do**
  for LS-Interval **do**
    for each sub-population **do**
      for depth of n generations **do**
        i) Create new individuals using genetic operators
        ii) Place new individuals in respective sub-population
      end for
    end for
  end for

  – Concatenate the best individuals from each sub-population into meme M
  – Encode M into recurrent network
  for LSI on local search population (l generations) **do**
    – crossover-based local search
    –restart if converged
  end for

  i) Decompose the refined individuals for respective sub-population
  ii) Replace the worst individuals of the respective sub-populations with the decomposed individual
end while

The *meme* is the individual that goes through local search. The details of the memetic cooperative neuro-evolution method is given in Algorithm 1. The algorithm assumes that it has been given the best parameters for the evolutionary algorithm such as the sub-population size, crossover and mutation rate.

The algorithm begins by encoding the recurrent neural network into the sub-population according to the respective cooperative coevolution encoding scheme. The specific encoding scheme for this work is neuron-based sub-population (Chandra et al. 2011c) for training recurrent networks.

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 sub-populations. Each sub-population is evolved for a depth of search of n generations in a round-robin fashion and the cycle is completed.

This process is repeated according to the local search interval. After the specified LS-Interval has been reached, the

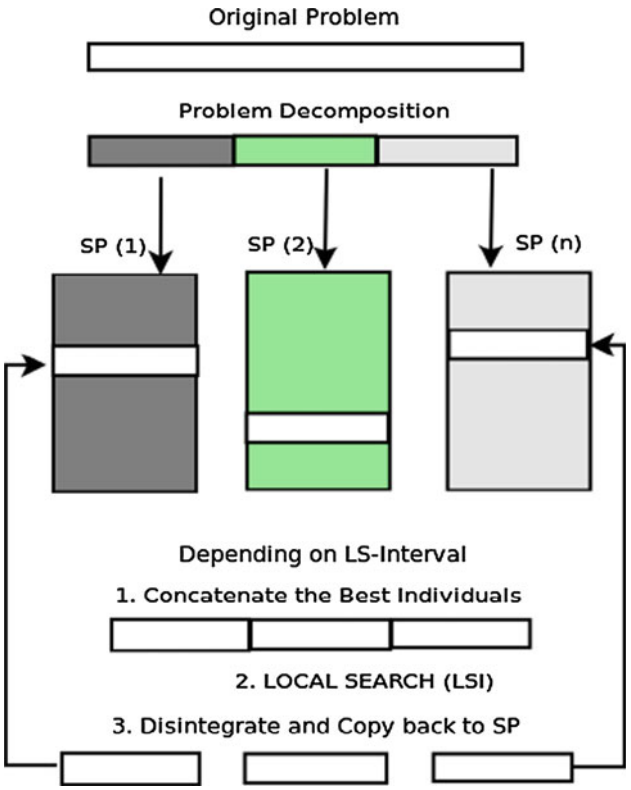


Fig. 3 The memetic cooperative coevolution method for training recurrent networks (Chandra et al. 2012b)

best individuals from all the sub-populations are concatenated into a meme which is further refined as shown in Fig. 3. The meme replaces the weakest individual in the local search population. The meme is then refined using the local search population for a given number of generations as defined by the LSI (l generations).

The refined meme is then disintegrated and copied to the respective sub-populations. The refined meme replaces the weakest individual in each of the sub-populations. Note that even if the refined meme is not improved, it replaces the weakest individuals as it may have features that will be used later in evolution. However, the best memes in the local search population are always retained. Although crossover-based local search is used as the designated method, the framework can employ any other local search method. Some of the components of the proposed method are discussed in the following subsections.

3.1 Initialisation

The feature of the cooperative coevolution sub-populations is to promote diversity (Potter and De Jong 2000). The local search population provides intensification. All the individuals of the respective sub-population are initialised with random real values. Each individual chromosome is then concatenated with the best individuals of the rest of the sub-populations and then encoded into a neural network and evaluated as done in Potter and De Jong (2000), Chandra et al. (2011c).

3.2 Diversity in competition

Cooperative coevolution naturally retains diversity through the use of sub-populations, where mating is restricted to the sub-populations and cooperation is mainly by collaborative fitness evaluation (Potter and Jong 1994; Potter and De Jong 2000). Since selection and recombination is restricted to a sub-population, the new solution will not have features from the rest of the sub-populations; therefore cooperative coevolution produces more diverse population when compared to a standard evolutionary algorithm with a single population.

The proposed memetic cooperative coevolution method employs competition in the local search population. The meme is refined in a population (different set of individuals) that is isolated from the sub-populations and then later the best individual (meme) is added to the sub-populations of cooperative coevolution which ensures higher level of diversity.

3.3 Fitness evaluation of subcomponents

The fitness of a given individual in a sub-population is obtained by combining it with the best individuals from the rest of the sub-populations. The concatenated individuals are encoded into the neural network where the fitness is evaluated and assigned back to the given individual. This method has been used to train cascade networks on the two-spirals problem and has shown to learn the task with smaller networks when compared to the cascade correlation learning architecture (Potter and De Jong 2000).

3.4 Local refinement using crossover-based local search

The crossover-based local search employs a population of few individuals, which is also referred as the local search population. The use of evolutionary algorithms for local search has been effective (Kazarlis et al. 2001; Lozano et al. 2004b; Molina et al. 2010). In the XLCC, the generalised generation gap with parent-centric crossover (G3-PCX) evolutionary algorithm (Deb et al. 2002) with a small population size is used as the evolutionary algorithm for crossover-based local search. The G3-PCX is also used as the evolutionary algorithm for the sub-populations of cooperative coevolution. The parent-centric crossover operator of the G3-PCX has features to provide good local search, therefore, it needs large population size (of more than 90) even for small 2 dimensional problems as discussed in Pošik (2009). A small population size for the G3-PCX will ensure that it becomes local search intensive and therefore it is used as a local search method.

The individuals in the population of the crossover-based local search are randomly seeded in the beginning of the evolutionary process. The cooperative coevolution sub-populations are seeded at the same time. During the evolutionary process, the cooperative coevolution sub-populations transfer the meme, which is the best solution, to the crossover-based local search population. This is done by concatenating the best solutions from all the sub-populations as shown in Fig. 3. This transfer is also dependent on the local search interval. Once the meme is transferred, the local search population is evolved according to the local search intensity. This population consists of the current meme and other candidate solutions left from the previous time when this population was used.

Once the local search population has been evolved according to the local search intensity, the best solution is transferred to the sub-populations of the cooperative coevolution. The remaining individuals in the local search population are kept and used in future local search evolution. This is done in order to maintain diversity, i.e. these individuals can be used to produce more fit offspring with the next meme that contains the best solution from cooperative coevolution.

A restart scheme is used when the local search population contains solutions that are similar to each other indicating local convergence. The population restart scheme is implemented by keeping the strongest individual aside and then initialisation the rest of the individuals with random numbers from a distribution. Afterwards, the strongest individual is added back to the local search population.

3.5 Other local search methods

Meta-Lamarckian learning can also be used in this framework. In meta-Lamarckian learning, several local searches can be employed and the suitable memes are chosen from the pool of local searchers as discussed in Ong and Keane (2004). However, for the case of neural network training, where function evaluation is costly, employing multiple local searches may not be practical for the given problem. Nevertheless, it may be suitable for problems where function evaluation is not very costly.

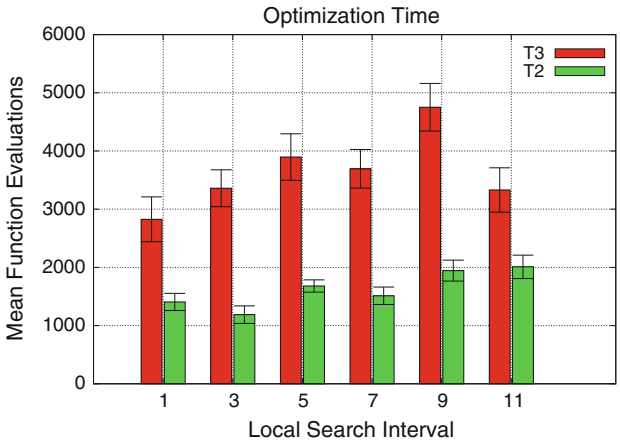
4 Simulation and analysis

This section presents an experimental study on the memetic cooperative coevolution method applied for training recurrent neural networks. The training and testing dataset is used from Chandra et al. (2011a,c, 2012a). We used grammatical inference problems from the Tomita language (Tomita 1982). We used Tomita 1 (T1), Tomita 2 (T2), Tomita 3 (T3), and Tomita 4 (T4). We also used a fuzzy finite automata (FFA) which has also been used to train Elman recurrent networks (Chandra et al. 2011a,c, 2012a). Neural level problem decomposition (Chandra et al. 2011c) shown in Fig. 1 is used in all the experiments.

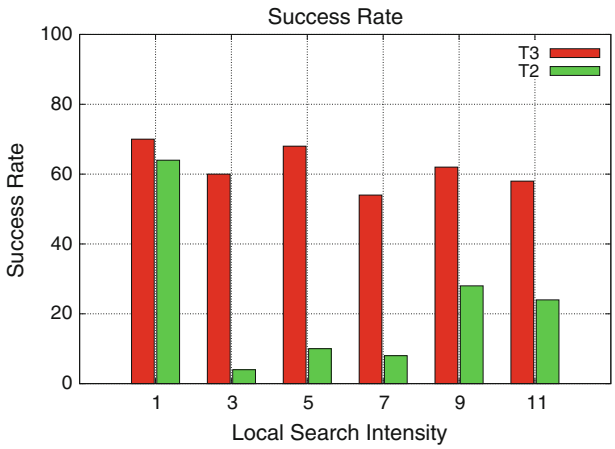
We report the training behaviour of the respective algorithms in terms of the function evaluations and the success rate. A run is successful when the desired solution is found before reaching the maximum training time. This determines the success rate. The goal of each algorithm is to obtain a high success rate with the least number of average function evaluations.

4.1 Local search intensity and interval

The FFA problem employs 5 neurons in the hidden layer. 2 neurons in the hidden layer for T2 problem and 3 neurons for the hidden layer for T3 and T4 problems are used. The maximum number of function evaluations in T2, T3 and T4 problems are 2000, 5000 and 5000, respectively. The FFA

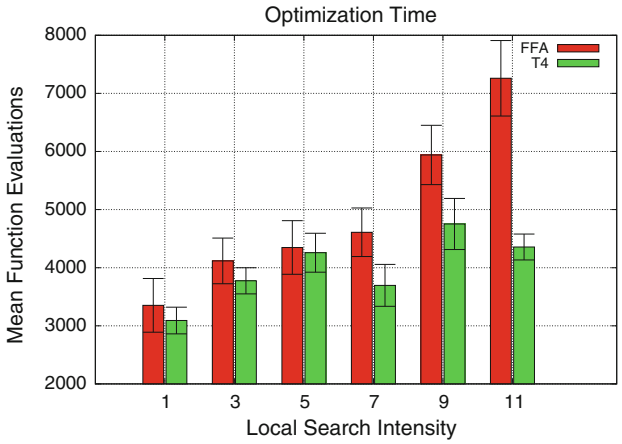


(a) Optimisation Time in Function Evaluations for evaluating the LS-Interval

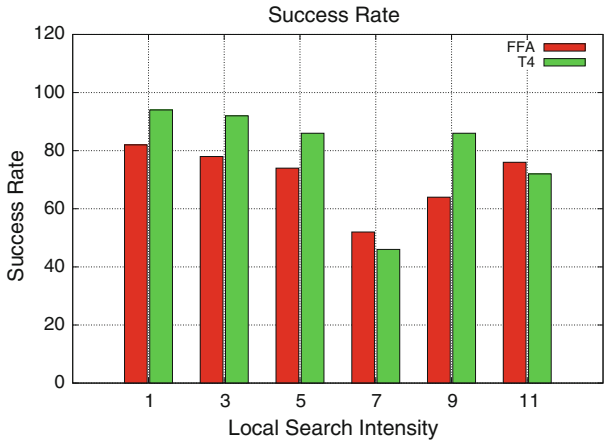


(b) Success Rate for evaluating the LS-Interval

Fig. 4 The evaluation of the LS-Interval for the T2 and T3 grammatical inference problems. The LSI of 8 generations is fixed in all problems. The frequency of 1 shows the best success rate and least number of function evaluations for all problems

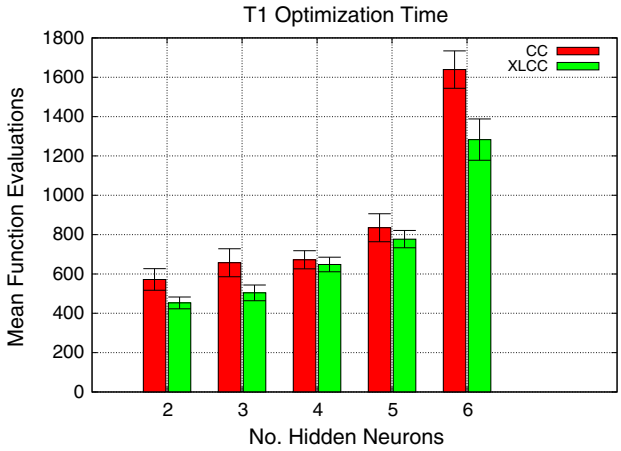


(a) Optimisation Time in Function Evaluations for evaluating the LS-Interval

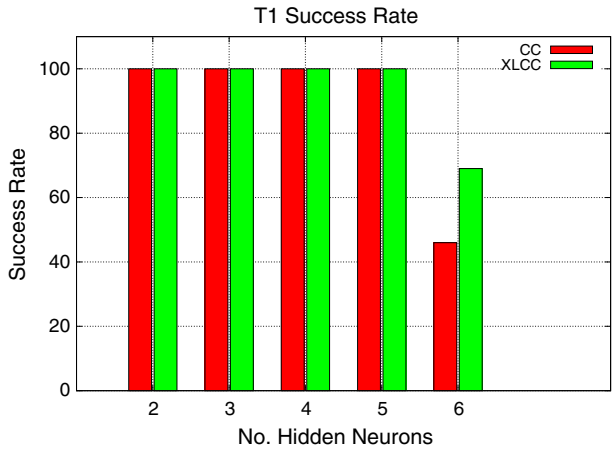


(b) Success Rate for evaluating the LS-Interval

Fig. 5 The evaluation of the LS-Interval for the FFA and T4 grammatical inference problems. The LSI of 8 generations is used as a fixed parameter in all problems. The interval of 1 shows the best success rate and least number of function evaluations for all problems



(a) Optimisation Time given by Mean Function Evaluations



(b) Success Rate

Fig. 6 The T1 problem

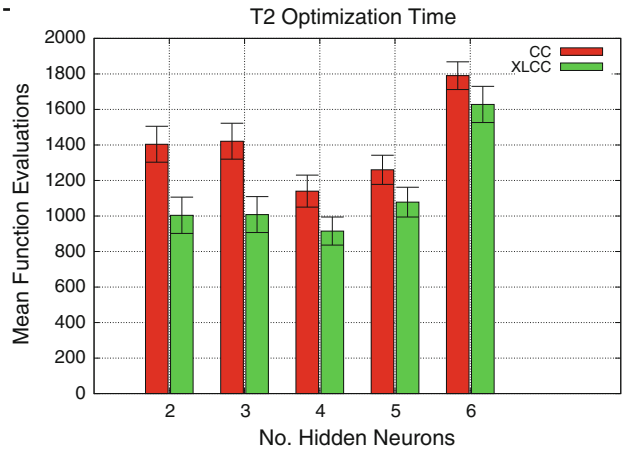
problem has 7000 has the maximum number of function evaluations. This set up has also been used in previous work Chandra et al. (2011a,c).

Figures 4 and 5 gives the results which shows the behaviour of XLCC on different LS-Interval for the 4 problems. 95 % confidence interval for 100 experiments is shown as error bars in the histograms. A good performance is given when the least optimisation time is used with the highest success rate. The fixed LSI of 8 generations is used. The LS-Interval of 1 gives the best performance in terms of the optimisation time (least function evaluations) as shown in Fig. 4a with better success rates as shown in Fig. 4b for the T2 and T3 problems. The LS-Interval of 3 for the T2 problem shows better optimisation time, however, it has poor success rate and therefore, LS-Internal of 1 has better performance. It is seen that the optimisation time and success rate deteriorates as the LS-Interval is increased for both the problems. The LS-Interval higher than 1 requires more time in terms of the number of function evaluations.

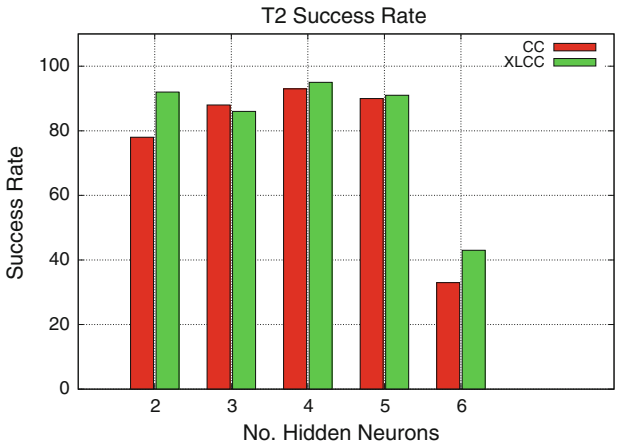
In Fig. 5, the LS-Interval is evaluated for the FFA and T4 problems. In both problems, the LS-Interval of 1 gives the best performance in terms of the optimisation time and the success rate. The performance deteriorates as the LS-Interval is increased.

4.2 Adaptive local search intensity

In the previous subsection, it has been established that the local search interval of 1 gives the best results. It is important to use the right local search intensity that may vary according to the problem. In the evolutionary process, global search is useful in the initial stage and local search in the later stages. The local search intensity should increase during the later stages in order to provide more emphasis for intensification. An adaptive method for determining the local search intensity is shown in Eq. 2

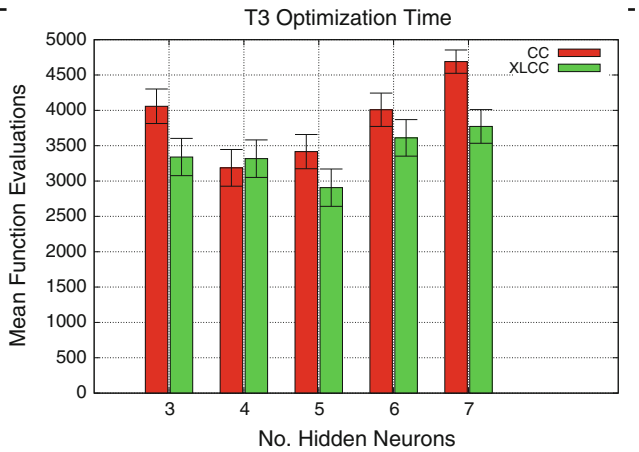


(a) Optimisation Time given by Mean Function Evaluations

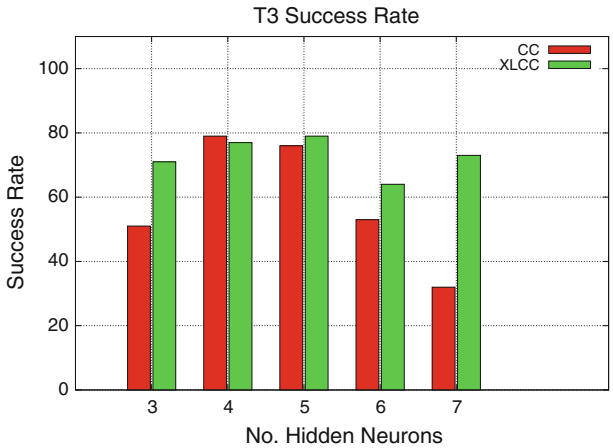


(b) Success Rate

Fig. 7 The T2 problem



(a) Optimisation Time given by Mean Function Evaluations



(b) Success Rate

Fig. 8 The T3 problem

$$LSI = 1 + \left(\frac{t}{m} * k\right) \tag{2}$$

where, t is the total number of function evaluations, m is the maximum number of function evaluations and k is a constant which specifies the maximum intensity of local search to be done in the final stages. This heuristic ensures that the intensity of local search increases with the number of function evaluations. We use $k = 30$ for all the problems in this study. The adaptive local search intensity gave good performance for training feedforward neural networks for pattern classification (Chandra et al. 2012b).

In these experiments, the maximum number of function evaluation for T1 and T2 problems are 2000. T3, T4 and T5 problems use 5000. All problems use different number of hidden neurons to test robustness.

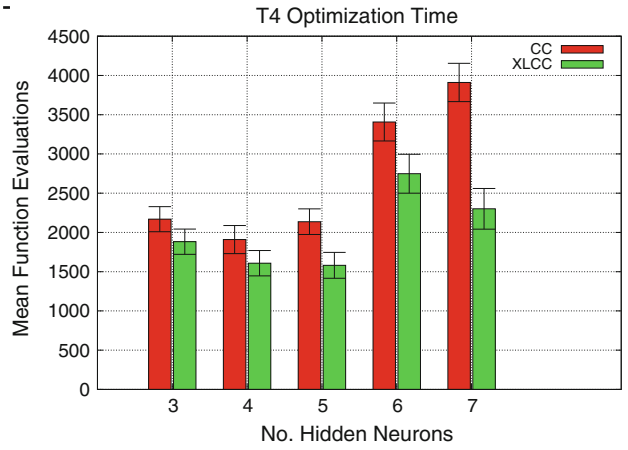
The results are shown in Figs. 6, 7, 8, 9, 10. In T1 problem shown in Fig. 6, XLCC performs better than CC in most cases. In T2 problem shown in Fig. 7, XLCC performs better for most cases. In the case of 2 neurons in the hidden layer, the optimisation time of XLCC is better, however the success rate is a bit weaker when compared to CC. In T3 problem shown in Fig. 8, XLCC performs better than CC in most cases, except for 4 hidden neurons where CC is slightly better. In the T4 and FFA problems in Figs. 9 and 10, respectively, XLCC shows better performance in all the cases.

The comparison of XLCC with standalone cooperative coevolution (CC) shows that XLCC has given better overall performance in terms of the optimisation time given by the number of function evaluations and the success rate.

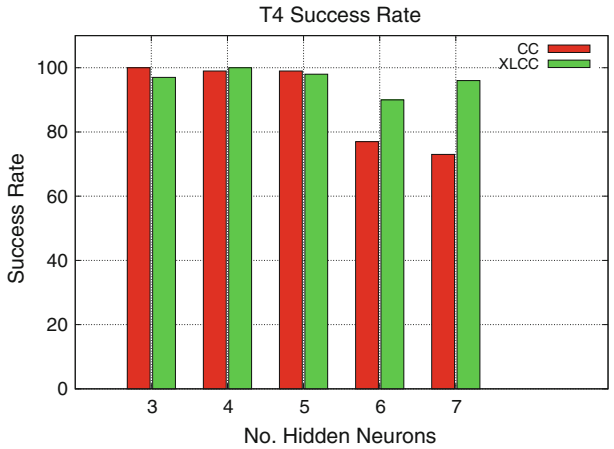
4.3 Discussion

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

In general, comparison of XLCC with CC shows improved performance in all most cases. This indicates that it is important to employ local search in cooperative coevolution for training recurrent neural networks. The results have clearly

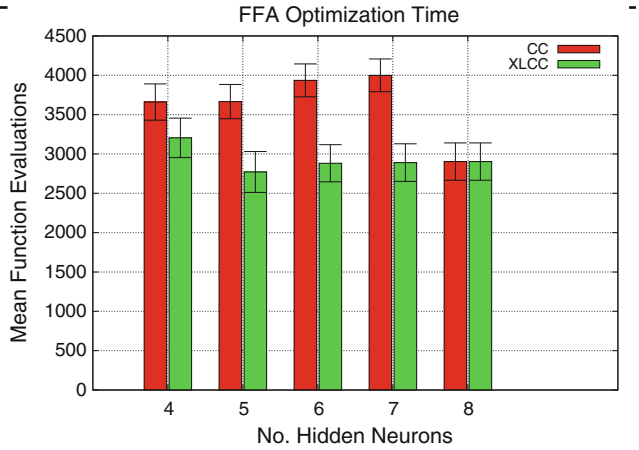


(a) Optimisation Time given by Mean Function Evaluations

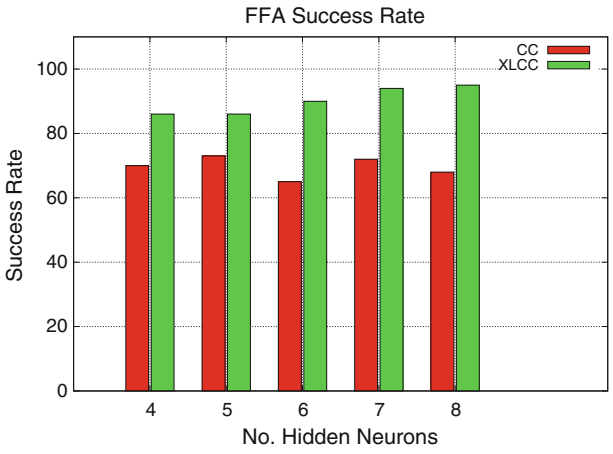


(b) Success Rate

Fig. 9 The T4 problem



(a) Optimisation Time given by Mean Function Evaluations



(b) Success Rate

Fig. 10 The FFA problem

shown that the adaptive depth of search has been beneficial as it gives better performance when compared to cooperative coevolution alone in terms of optimisation time and success rate. The proposed memetic cooperative coevolution method performs well in terms of robustness as its performance is not deteriorated different number of hidden neurons.

Co-adaptation is necessary in cooperative coevolution, especially in the case where the problem has difficulty in decomposition. It is difficult to decompose neural networks into subcomponents as the interaction between the synapses depends on the network architecture and the nature of the problem, i.e training data (Chandra et al. 2012c). The local search population has also provided features of co-adaptation between the several sub-populations of cooperative coevolution. This population provides the means for selected individuals to be exchanged with different sub-populations using the crossover operation in the local search population. Moreover, the restart scheme in the local search population also provides features of adaptation when local minimum has been reached.

Although feedback connections are present for recurrent networks, the performance of XLCC for training them is similar when compared to feedforward networks (Chandra et al. 2012b). The similarity in performance is in terms of the reduced optimisation time when compared to CC and better guarantee for convergence by XLCC when compared to CC. Moreover, XLCC is also better in terms of scalability, i.e, the adaptability of the algorithm given different number of hidden neurons.

The LS-Interval of 1 has given the best performance for recurrent networks in this study and feedforward network in our previous work (Chandra et al. 2012b). XLCC is purely an evolutionary computation method that does not rely on gradient information. It is appropriate for applying neural networks for control problems where gradient information is not easily available.

5 Conclusions and future work

This paper applied an established memetic cooperative coevolution method for training recurrent neural networks for a set of learning problems given by deterministic and fuzzy finite state automata. The relationship between the

local search interval and local search intensity was first established and then the method was used for training recurrent networks given different numbers of hidden neurons that reflected in terms of robustness.

The results have shown improved performance in terms of optimisation time and guarantee of convergence which opens the road for further research in using other local refinement procedures with cooperative coevolution.

In future work, other local search methods can replace or be added with the crossover-based local search for local refinement. Backpropagation-through-time can be used as an additional local search method to incorporate gradient information and enhance the evolutionary search process. The memetic cooperative coevolution method can also be used to train other recurrent network architectures and also extended for global optimisation problems.

References

Acampora G, Cadenas J, Loia V, Ballester E (2011) Achieving memetic adaptability by means of agent-based machine learning. IEEE Trans Indus Inform 7(4):557–569

Acampora G, Gaeta M, Loia V (2011) Combining multi-agent paradigm and memetic computing for personalized and adaptive learning experiences. Comput Intell 27(2):141–165

Acampora G, Loia V, Salerno S, Vitiello A (2012) A hybrid evolutionary approach for solving the ontology alignment problem. Int J Intell Syst 27(3):189–216

Chandra R, Frean M, Zhang M (2010) An encoding scheme for cooperative coevolutionary neural networks. In: 23rd Australian joint conference on artificial intelligence. Lecture notes in artificial intelligence. Springer, Adelaide, Australia, in Press

Chandra R, Frean M, Zhang M (2011a) A memetic framework for cooperative coevolution of recurrent neural networks. In: The 2011 international joint conference on neural networks (IJCNN), pp 673–680

Chandra R, Frean M, Zhang M (2011b) Modularity adaptation in cooperative coevolution of feedforward neural networks. In: The 2011 international joint conference on neural networks (IJCNN), pp 681–688

Chandra R, Frean M, Zhang M, Omlin CW (2011c) Encoding subcomponents in cooperative co-evolutionary recurrent neural networks. Neurocomputing 74(17):3223–3234

Chandra R, Frean M, Zhang M (2012a) Adapting modularity during learning in cooperative co-evolutionary recurrent neural networks. Soft Comput Fusion Found Methodol Appl 16(6):1009–1020

Chandra R, Frean M, Zhang M (2012b) Crossover-based local search in cooperative co-evolutionary feedforward neural networks. Appl Soft Comput 12(9):2924–2932

Chandra R, Frean M, Zhang M (2012c) On the issue of separability for problem decomposition in cooperative neuro-evolution. Neurocomputing 87:33–40

Deb K, Anand A, Joshi D (2002) A computationally efficient evolutionary algorithm for real-parameter optimization. Evol Comput 10(4):371–395

Elman JL (1990) Finding structure in time. Cogn Sci 14:179–211

Giles CL, Horne BG, Lin T (1995) Learning a class of large finite state machines with a recurrent neural network. Neural Netw 8(9):1359–1365

Glover FW, Kochenberger GA (2003) Handbook of metaheuristics. Springer, Berlin

Gomez F, Mikkulainen R (1997) Incremental evolution of complex general behavior. Adapt Behav 5(3–4):317–342

Gomez F, Schmidhuber J, Miikkulainen R (2008) Accelerated neural evolution through cooperatively coevolved synapses. J Mach Learn Res 9:937–965

Gomez FJ (2003) Robust non-linear control through neuroevolution. Technical Report AI-TR-03-303, PhD thesis, Department of Computer Science, The University of Texas at Austin

Haykin S, Principe J, Sejnowski T, McWhirter J (2006) New directions in statistical signal processing: from systems to brain. MIT Press, Cambridge

Kazarlis SA, Papadakis SE, Theocharis IB, Petridis V (2001) Microgenetic algorithms as generalized hill-climbing operators for ga optimization. IEEE Trans Evolut Comput 5(3):204–217

Kolen J, Kremer S (2001) A field guide to dynamical recurrent networks. IEEE Press, Piscataway, NJ, USA

Kremer S (1995) On the computational power of elman-style recurrent networks. IEEE Trans Neural Netw 6(4):1000–1004

Lin C-J, Chen C-H, Lin C-T (January 2009) A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications. Trans Syst Man Cyber Part C 39:55–68

Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-coded memetic algorithms with crossover hill-climbing. Evol Comput 12:273–302

Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-coded memetic algorithms with crossover hill-climbing. Evol Comput 12(3):273–302

Manolios P, Fanelli R (1994) First-order recurrent neural networks and deterministic finite state automata. Neural Comput 6(6):1155–1173

Medsker L, Jain L (1999) Recurrent neural networks: design and application, computer intelligence. CRC Press, Florida, USA

Molina D, Lozano M, Garca-Martnez C, Herrera F (2010) Memetic algorithms for continuous optimisation based on local search chains. Evol Comput 18(1):27–63

Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Tech Rep

Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical Report 826, Caltech Concurrent Computation Program

Moscato P (2003) A gentle introduction to memetic algorithms. In: Handbook of Metaheuristics. Kluwer Academic Publishers, Dordrecht, pp 105–144

Nguyen QH, Ong Y-S, Lim MH (2009) A probabilistic memetic framework. IEEE Trans Evolut Comput 13(3):604–623

Ong YS, Keane A (2004) Meta-lamarckian learning in memetic algorithms. IEEE Trans Evolut Comput 8(2):99–110

Pham DT, Karaboga D (1999) Training elman and jordan networks for system identification using genetic algorithms. Artif Intell Eng 13(2):107–117

Potter MA, De Jong KA (2000) Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evol Comput 8(1):1–29

Potter MA, Jong KAD (1994) A cooperative coevolutionary approach to function optimization. Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature. Springer, London, UK, PPSN III, pp 249–257

Pošik P (2009) Bbob-benchmarking the generalized generation gap model with parent centric crossover. In: Proceedings of the 11th annual conference companion on genetic and evolutionary computation conference: late breaking papers. GECCO '09, pp 2321–2328

Robinson T (1994) An application of recurrent nets to phone probability estimation. IEEE Trans Neural Netw 5:298–305

Seyab RA, Cao Y (2008) Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation. J Process Control 18(6):568–581

Smith J (2007) Coevolving memetic algorithms: a review and progress report. IEEE Trans Syst Man Cybern Part B Cybern 37(1):6–17

Tang J, Lim MH, Ong YS (2007) Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. Soft Comput 11(9):873–888

Tomita M (1982) Dynamic construction of finite automata from examples using hill-climbing. Proceedings of the fourth annual cognitive science Conference. MI, Ann Arbor, pp 105–108

Watts DJ (1999) Small worlds: The dynamics of networks between order and randomness. Princeton University Press, Princeton

Zhou Z, Ong YS, Lim MH, Lee BS (2007) Memetic algorithm using multi-surrogates for computationally expensive optimization problems. Soft Comput 11(10):957–971