Evolutionary multi-task learning for modular training of feedforward neural networks

Rohitash Chandra, Abhishek Gupta, Yew-Soon Ong, and Chi-Keong Goh

Rolls Royce @ NTU Corporate Lab
Nanyang Technological University
Nanyang View, Singapore.

Abstract. Multi-task learning enables learning algorithms to harness shared knowledge from several tasks in order to provide better performance. In the past, neuro-evolution has shown promising performance for a number of real-world applications. Recently, evolutionary multi-tasking has been proposed for optimisation problems. In this paper, we present a multi-task learning for neural networks that evolves modular network topologies. In the proposed method, each task is defined by a specific network topology defined with a different number of hidden neurons. The method produces a modular network that could be effective even if some of the neurons and connections are removed from selected trained modules in the network. We demonstrate the effectiveness of the method using feedforward networks to learn selected n-bit parity problems of varying levels of difficulty. The results show better training and generalisation performance when the modules for representing additional knowledge are added by increasing hidden neurons during training.

Key words: Evolutionary multitasking, neuro-evolution, modular design, multi-task learning

1 Introduction

Neuro-evolution employs evolutionary algorithms for training neural networks [1] which can be classified into direct [1, 2], and indirect encoding strategies [3]. In direct encoding, every connection and neuron is specified directly and explicitly in the genotype [1, 2]. Direct encoding has also been used in the evolution of feedforward networks for pattern recognition problems using conventional evolutionary algorithms [4], memetic based approaches [5] and cooperative coevolution [6–8].

In indirect encoding, the genotype specifies rules or some other structure for generating the network. Neuro-evolution of augmenting topologies (NEAT) has been a popular indirect encoding that begins evolution with the simplest network topology and adapts nodes and weights together during evolution [3]. Performance of direct and indirect encodings vary for specific problems, while indirect encodings seem very intuitive and have biological motivations, they in several cases have shown not to outperform direct encoding strategies [7, 9]. Neuro-evolution has also been applied to evolve modular neural networks [10].

Modular neural networks are motivated from repeating structures in nature [10]. They were introduced for for visual recognition tasks that were trained by genetic algorithms and produced generalisation capability [10]. More recently, a modular neural network was presented where the performance and connection costs were optimised through neuro-evolution which achieved better performance when compared to fully connected neural networks [11]. They have also been designed with the motivation to learn new tasks without forgetting old ones [12]. It was shown that modular networks learn new tasks faster from knowledge of previous tasks. Modular neural network architectures have been beneficial for hardware implementations [13]. In particular, they enable smaller networks to be used as building blocks for a larger network. Such transfer of knowledge from simpler to progressively more complex tasks can be achieved by the process of multi-task learning [14].

The notion of multi-task learning has been prevalent in the machine learning community for at least the past two decades [14]. The main stimulus for the approach has been the potential for exploiting relevant information available in related tasks by simultaneous learning using a shared representation. Keeping this motivation in mind, we hereafter extend this idea to neuro-evolution where the implicit parallelism of population-based search is fully unleashed via *evolutionary multitasking* - a novel paradigm that has only recently been conceived [15–17]. Multitasking, from the standpoint of *optimization*, has shown to facilitate autonomous knowledge exchange (in the form of implicit genetically encoded information transfer) between essentially self-contained (but possibly similar) tasks that are processed (evolved) concurrently in a unified solution representation space, thereby often leading to improved convergence characteristics.

In this paper, we present multi-task learning for neural networks that evolves modular network topologies. In the proposed method, each of the tasks is defined by specific network topologies that can also be viewed as modules which feature building blocks for transfer. The method produces a modular network that is effective or feasible even if some of the neurons and connections are removed from selected modules in the trained network. We demonstrate the effectiveness of the method using feedforward neural networks (FNNs) that learn different instances of the n-bit parity problem.

The rest of the paper is organised as follows. Section 2 presents the proposed method and Section 3 presents experiments and results. Section 4 concludes the paper with a discussion of future work.

2 Evolutionary Multi-task Learning

Although neural networks have been successfully applied to several real-world problems, their design of topology is typically based on trial-and-error which is often time consuming. While neuro-evolution has been popular, the potential for multi-task learning though neuro-evolution has not been explored.

Essentially, neuro-evolution employs an evolutionary algorithm (EA) that features operators such as selection, crossover and mutation for evolving the weights of the feedforward network. The EA is composed of a population of p individuals (chromosomes) that are real-coded to represent the weights and bias in the network. The individuals are mapped into the network using direct encoding techniques where all the weights are encoded in a consecutive order, hence each individual consists of total number of weights and biases in the network.

The proposed method, where multitasking is used for neuro-evolution considers different tasks as neural network topologies defined by different number of hidden neurons. Since, we are employing direct encoding for weight representation in the evolutionary algorithm, different tasks in the multi-tasking method results in varied length real-parameter chromosomes in the evolutionary algorithm. Hence, we employ the strategies for creating new offspring using genetic operators proposed in evolutionary multi-tasking [15]. We focus on tasks as network topologies that are applied to the same learning problem.

Alg. 1 Multi-Tasking Neuro-Evolution for FNNs

```
Step 1: Define different tasks given by number of hidden neurons
Task 1 ($p$ hidden neurons); Task 2 ($q$ hidden neurons); Task 3 ($r$ hidden neurons)
Step 2: Initialise popsize individuals in the unified search space
i) Randomly associate every individual with any one of Task (1, 2, or 3)
ii) Evaluate individuals for their associated task only
for each generation until termination do
    i) Select and create new offspring via crossover and mutation
    ii) Associate each offspring with any one of Task (1, 2, or 3) via imitation
    ii) Evaluate offspring for their associated task only
    iii) Select elitist popsize/3 indvduals from each of Task (1, 2, and 3) for next
    generation
end for
```

Algorithm 1 gives details for the proposed evolutionary multi-task learning method. Without loss of generality, we assume there exists three tasks at a time. The algorithm follows a similar workflow as the multi-factorial evolutionary algorithm proposed in [15].

The critical step for successful evolutionary multi-task learning is the formulation of the unified solution representation scheme encompassing the search spaces of all tasks. Such unification can be achieved in a straightforward manner in a multi-dimensional real space for real-parameter optimization (as is the case with training neural network weights). The main challenge is with regard to accommodating for the heterogeneity in search space dimensionality of constitutive tasks. Accordingly, following the methodology proposed in [15], if the three tasks in Algorithm 1 possess dimensionality D_1, D_2, and D_3, respectively, then the dimensionality of the unified search space is given by $D_{multitask} = \max\{D_1, D_2, D_3\}$. Thus, a candidate solution in the unified space is characterized by a vector of $D_{multitask}$ elements. While evaluating an individual associated with the j^{th} task, we simply refer to (or extract) D_j relevant elements from the list of $D_{multitask}$ elements. The motivation behind employing such a search space merging scheme (instead of simplistic concatenation) is to facilitate the underlying EA in providing implicit information transfer across related tasks. This transfer, at least within the present framework, is achieved during the crossover operation as discussed in [16].

At this juncture, an appropriate choice of D_j elements to be extracted for the j^{th} task is crucial for effective multi-task learning. This must be customized by accounting for the properties of the multi-task learning instance at hand. For the present example, we assume p (hidden neurons in Task 1) $< q$ (hidden neurons in Task 2) $< r$ (hidden neurons in Task 3); which implies that $D_1 < D_2 < D_3$. Thus, we enforce that all weights connected to the first p hidden neurons (regardless of Task [1, 2 , or 3]) refer to the same $D_p = D_1$ elements among $D_{multitask}$. Similarly, all the weights connected to the next q - p hidden neurons (regardless of Task [2 or 3]) refer to the same $D_{q-p} = D_2$ - D_1 elements among $D_{multitask}$. Finally, all the weights connected to the last r - q hidden neurons in Task 3 refer to the last $D_{r-q} = D_3$ - D_2 elements among $D_{multitask}$. Notice that in the proposed procedure $D_p + D_{q-p} + D_{r-q} = D_{multitask}$. The encoding process is further illustrated in Figure 1.

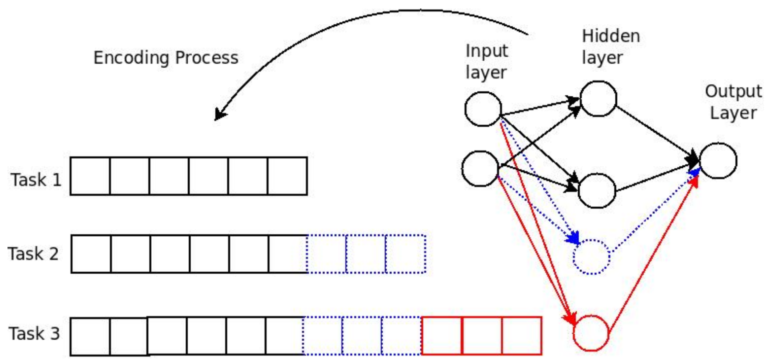

Fig. 1. Encoding FNN as tasks in evolutionary multi-task learning. Note that the colours associated with the synapses in the network are linked to their encoding that are given as different tasks. Task 1 employs a network topology with 2 hidden neurons while the rest of the tasks add extra hidden neurons to it.

To conclude the description of the algorithm, we highlight the association of every individual with *any one* task in the multi-task learning environment. Doing so primarily leads to a saving in the computational cost as evaluating every individual exhaustively for every task is likely to be expensive. While in the initial population the associations are randomly assigned (whilst ensuring uniform representation for all tasks), an assignment strategy based on the memetic concept of *vertical cultural transmission* [18] is adopted in subsequent generations for the genetically modified offspring. In particular, it is prescribed that an offspring randomly *imitates* the association of any one of the parents from which it is created. For further details, refer to [15].

2.1 n-bit Parity Problem

The n-bit parity problem essentially is defined as the number of even parity bits in a binary string. Typically, n is the size of the string and all the possible permutations are used to make a dataset.

The n-bit parity problem has been used in the literature to demonstrate the effectiveness of a particular neural network architecture [19] or training algorithm [20]. For instance, the n-bit parity problem was solved using a neural network that allowed direct connections between the input layer and the output layer and trained with linear programming [19]. The effectiveness of certain neuro-evolution methods has also been demonstrated using n-bit problem [21, 22, 20].

3 Experiments and Results

This section presents the experimental results for evolutionary multi-task learning (EMTL) of feedforward network topologies for the n-bit parity problem. Although 3 or 4 bit parity problems have also been used in the literature [21, 22], we specifically chose 4, 6 and 8-bit parity problems in order to demonstrate the effectiveness of the proposed method in order to cater for increasing levels of difficulty.

3.1 Design of Experiments

We design the experiments where we compare the performance for the evolutionary single-task learning (ESTL) approach given different number of hidden neurons with EMTL.

The ESTL is implemented through an EA that employs a population size (*popsize*) of 30 individuals. Accordingly, a population of 90 individuals is employed for EMTL which executes three self-contained tasks at once. The algorithm terminates once a total of at least 30 000 function evaluations are completed for each task. Furthermore, note that identical crossover and mutation operators are employed for both methods. In particular, a simulated binary crossover (SBX) operator (with distribution index of 2) [23] and polynomial mutation (with distribution index of 5) are used [24]. The choice of low distribution indices encouraging enhanced exploration is preferred as we employ an elitist selection strategy ensuring preservation of high quality genetic material.

The mean squared error (MSE) given in Equation 1 and the classification performance are used to evaluate the performance of the respective methods.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1}$$

where y_i and $\hat{y}_i$ are the observed and predicted output, respectively. N is the length of the data.

3.2 Results

The results for 4, 6 and 8 bit problems that compare EMTL with ESTL are given in Table 1 - Table 6, for MSE and classification performance, respectively.

In Tables 1, 3 and 5, EMTL achieves better performance than ESTL for all the cases for the respective problems. The same trend is also given for the classification performance in Tables 2, 4 and 6 . The results clearly demonstrate that the EMTL is better than ESTL given that the overall optimisation time is the same for different strategies (given by number of hidden neurons) for all the respective problems.

Table 1. MSE (std-dev) of EMTL vs ESTL for the 4-bit problem

Method	Task 1	Task 2	Task 3
EMTL(4, 5, 6)	0.0653 (0.0591)	0.0472 (0.0433)	0.0361 (0.0326)
ESTL (4, 5, 6)	0.1557 (0.0364)	0.1020 (0.0232)	0.0699 (0.0285)
EMTL(6, 7, 8)	0.0071 (0.0219)	0.0058 (0.0178)	0.0043 (0.0132)
ESTL(6, 7, 8)	0.0699 (0.0285)	0.0427 (0.0336)	0.0325 (0.0309)

Table 2. Classification performance (std-dev) of EMTL vs ESTL for the 4-bit problem

Method	Task 1	Task 2	Task 3
EMTL(4, 5, 6)	86.6667% (3.1714)	90.4167% (4.2590)	95.4167% (5.4272)
ESTL (4, 5, 6)	73.3333% (15.3912)	87.0833% (6.1267)	90.4167 % (6.5104)
EMTL(6, 7, 8)	91.8750 % (4.6857)	94.3750 % (3.0041)	98.1250% (2.9131)
ESTL(6, 7, 8)	90.4167 % (6.5104)	94.1667 % (4.9057)	94.7917 % (5.2119)

Table 3. MSE (std-dev) of EMTL vs ESTL for the 6-bit Problem

Method	Task 1	Task 2	Task 3
EMTL(4, 5, 6)	0.0977 (0.0240)	0.0789 (0.0264)	0.0639 (0.0224)
ESTL (4, 5, 6)	0.1343 (0.0392)	0.1153 (0.0235)	0.0949 (0.0254)
EMTL(6, 7, 8)	0.0642 (0.0221)	0.0459 (0.0148)	0.0492 (0.0269)
ESTL(6, 7, 8)	0.0949 (0.0254)	0.0695 (0.0304)	0.0643 (0.0272)

Figure 2 and 3 show the convergence trend for the respective strategies for the 8-bit parity problem. In the case, the mean MSE for 30 experimental runs is shown at different stages of evolution. It is clear that EMTL converges with higher quality solutions.

3.3 Discussion

The experiments were designed to observe the behaviour of evolutionary multi-task learning when compared to single-task learning. We observed interesting results where the multi-task learning performed better. This is due to the way the problem was decomposed using a dynamic programming approach where the main task is divided into smaller tasks as building blocks that transfer to larger tasks given by larger network topologies. The results show that the building

Table 4. Classification performance (std-dev) of EMTL vs ESTL for the 6-bit problem

Method	Task 1	Task 2	Task 3
EMTL(4, 5, 6)	85.3125 % (8.8064)	90.5208% (6.1129)	93.1250 % (4.0538)
ESTL (4, 5, 6)	81.1458 % (10.0337)	83.4375 % (9.2174	86.1458 % (8.1214)
EMTL(6, 7, 8)	92.7604% (4.5597)	95.6250 % (2.6081)	95.0000 % (4.5211)
ESTL(6, 7, 8)	86.1458 % (8.1214)	92.0833 % (7.4533)	92.5521 % (4.7087)

Table 5. MSE (std-dev) of EMTL vs ESTL for the 8-bit problem

Method	Task 1	Task 2	Task 3
EMTL(5, 6, 7)	0.1108 (0.0249)	0.0949 (0.0257)	0.0861 (0.0232)
ESTL (5, 6, 7)	0.1349 (0.0279)	0.1223 (0.0279)	0.1269 (0.0322)
EMTL(7, 8, 9)	0.0843 (0.0274)	0.0661 (0.0178)	0.0618 (0.0161)
ESTL(7, 8, 9)	0.1269 (0.0322)	0.1093 (0.0275)	0.0936 (0.0217)

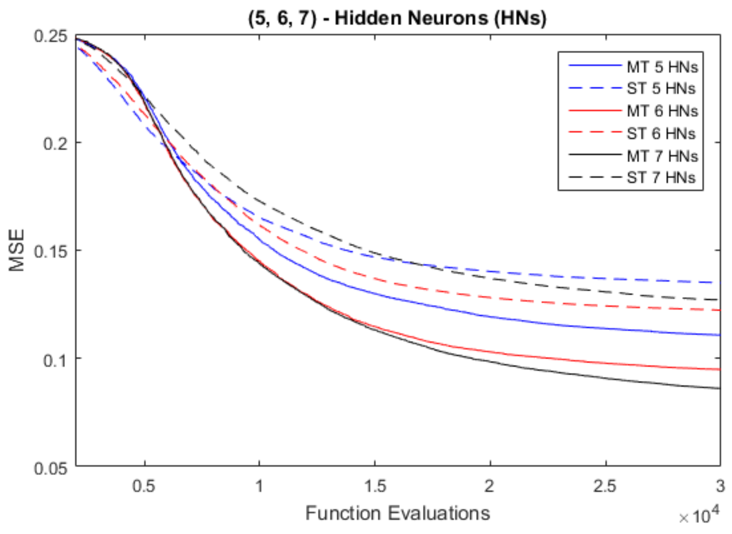

Fig. 2. Convergence trend for (5, 6, 7) neurons for the 8-bit problem. Note that the multi-task learning strategies (MT) with solid lines converge with higher solution quality when compared to single-task learning (ST) as the number of function evaluation increases.

blocks of knowledge in smaller tasks were transferred and effectively refined when presented with more weights and neurons in the larger tasks.

One can argue that the design of experiments are biased towards the multi-task learning approach as it was given more computational budget (90 000) when compared to single task learning approach (30 000). Note that this is because the multi-task learning approach is solving 3 tasks at the same time exhibiting parallelism, whereas, single task learning approaches the problem sequentially. Therefore, the total time for the respective learning methods is equal when we consider all the tasks. However, as shown in the results, multi-task learning

Table 6. Classification performance (std-dev) of EMTL vs ESTL for the 8-bit problem

Method	Task 1	Task 2	Task 3
EMTL(5, 6, 7)	86.5234 % (4.3712)	88.6198 % (4.6375)	89.0885 % (5.3569)
ESTL (5, 6, 7)	82.0313 % (6.4415)	81.9010 % (9.5784)	84.1406 % (6.1272)
EMTL(7, 8, 9)	90.5859 % (5.4334)	92.9427 % (3.2954)	93.5417 % (2.3810)
ESTL(7, 8, 9)	84.1406 % (6.1272)	85.7813 % (6.8659)	88.4115 % (5.0280)

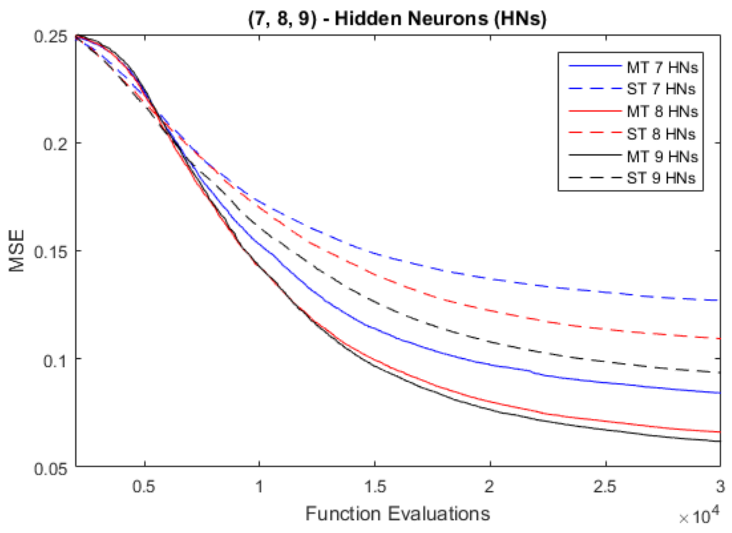

Fig. 3. Convergence trend for (7, 8, 9) neurons for the 8-bit problem. The multi-task learning strategies (MT) with solid lines converge with higher solution quality when compared to single-task learning (ST) as the number of function evaluation increases.

significantly outperforms the single task approach given an equal computational budget.

Modularity is enforced as the network with evolutionary multi-task learning was able to retain the knowledge of the previous tasks while learning the new tasks, i.e. tasks with smaller network topologies. In this way, multi-task learning does not forget or discard the knowledge learnt in smaller network topologies (modules) which can be useful if a lower number of neurons or connections are needed for hardware implementations. Moreover, multi-task learning delivers a neuro-evolution strategy which employs direct encoding along with the advantages of indirect encoding where the number of hidden neurons and associated connection are evolved during evolution.

In the case of hardware implementation [13], it is essential to have a modular trained neural network that is operational with a degree of error for *safe mode operation* when some of the neurons or links in selected modules are damaged during an event.

4 Conclusions and Future Work

We presented an evolutionary multi-task learning method that produced modular neural networks which have the property to retain knowledge in some of the modules when others are affected. This can be particularly helpful for hardware implementations where certain defects in the modules do not affect the entire knowledge representation and hence the network can operate in *safe mode*.

The results showed that the multi-task learning approach was helpful in achieving better solution quality when compared to a related method. This was due to the way the training was done where the larger problem was solved in parts using a dynamic programming approach.

In future work, the approach can be extended to tasks that consist of different problems rather than neural network topologies. There is scope for using other evolutionary techniques such as coevolution with application to problems that have building blocks or require multi-task learning. The design proposed method can also be helpful for hardware implementation.

References

1. 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.
2. D. E. Moriarty and R. Miikkulainen, “Forming neural networks through efficient and adaptive coevolution,” *Evolutionary Computation*, vol. 5, no. 4, pp. 373–399, 1997. [Online]. Available: http://www.mitpressjournals.org/doi/abs/10.1162/evco.1997.5.4.373
3. K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.
4. R. S. Sexton and R. E. Dorsey, “Reliable classification using neural networks: a genetic algorithm and backpropagation comparison,” *Decision Support Systems*, vol. 30, no. 1, pp. 11–22, 2000.
5. E. Cant-Paz and C. Kamath, “An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems,” *IEEE Transactions on systems, man and cybernetics-Part B: Cybernetics*, vol. 35, no. 5, pp. 915–933, 2005.
6. 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.
7. F. Gomez, J. Schmidhuber, and R. Miikkulainen, “Accelerated neural evolution through cooperatively coevolved synapses,” *J. Mach. Learn. Res.*, vol. 9, pp. 937–965, 2008.
8. R. Chandra, “Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 26, pp. 3123–3136, 2015.
9. V. Heidrich-Meisner and C. Igel, “Neuroevolution strategies for episodic reinforcement learning,” *Journal of Algorithms*, vol. 64, no. 4, pp. 152 – 168, 2009, special Issue: Reinforcement Learning.
10. B. L. Happel and J. M. Murre, “Design and evolution of modular neural network architectures,” *Neural Networks*, vol. 7, no. 67, pp. 985 – 1004, 1994, models of Neurodynamics and Behavior.

11. J. Clune, J.-B. Mouret, and H. Lipson, "The evolutionary origins of modularity," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 280, no. 1755, 2013.
12. K. O. Ellefsen, J.-B. Mouret, and J. Clune, "Neural modularity helps organisms evolve to learn new skills without forgetting old skills," *PLoS Comput Biol*, vol. 11, no. 4, pp. 1–24, 04 2015.
13. J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 13, pp. 239 – 255, 2010, artificial Brains.
14. R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.
15. A. Gupta, Y. S. Ong, and L. Feng, "Multifactorial evolution: Toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2016.
16. A. Gupta, Y.-S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Trans. Cybernetics*, Accepted, 2016.
17. Y.-S. Ong and A. Gupta, "Evolutionary multitasking: A computer science view of cognitive multitasking," *Cognitive Computation*, pp. 1–18.
18. X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 591–607, 2011.
19. D. Liu, M. E. Hohil, and S. H. Smith, "N-bit parity neural networks: new solutions based on linear programming," *Neurocomputing*, vol. 48, no. 14, pp. 477 – 488, 2002.
20. M. Mangal and M. P. Singh, "Analysis of pattern classification for the multidimensional parity-bit-checking problem with hybrid evolutionary feed-forward neural network," *Neurocomputing*, vol. 70, no. 79, pp. 1511 – 1524, 2007, advances in Computational Intelligence and Learning14th European Symposium on Artificial Neural Networks 200614th European Symposium on Artificial Neural Networks 2006.
21. S. Mirjalili, S. Z. M. Hashim, and H. M. Sardroudi, "Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11 125 – 11 137, 2012.
22. 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.
23. K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.
24. K. Deb and D. Deb, "Analysing mutation schemes for real-parameter genetic algorithms," *International Journal of Artificial Intelligence and Soft Computing*, vol. 4, no. 1, pp. 1–28, 2014.