Evolutionary multi-objective optimization of spiking neural networks

Yaochu Jin, Ruojing Wen, Bernhard Sendhoff

2007

Preprint:

This is an accepted article published in Artificial Neeural Networks-ICANN, 17. International Conference. The final authenticated version is available online at: https://doi.org/[DOI not available]

Evolutionary Multi-objective Optimization of Spiking Neural Networks

Yaochu Jin[1], Ruojing Wen[2], Bernhard Sendhoff[1]

[1] Honda Research Institute Europe
Carl-Legien-Str. 30, 63073 Offenbach, Germany
[2] Department of Computer Science, University of Karlsruhe
Zirkel 2, 76131 Karlsruhe, Germany

Abstract. Evolutionary multi-objective optimization of spiking neural networks for solving classification problems is studied in this paper. By means of a Pareto-based multi-objective genetic algorithm, we are able to optimize both classification performance and connectivity of spiking neural networks with the latency coding. During optimization, the connectivity between two neurons, i.e., whether two neurons are connected, and if connected, both weight and delay between the two neurons, are evolved. We minimize the the classification error in percentage or the root mean square error for optimizing performance, and minimize the number of connections or the sum of delays for connectivity to investigate the influence of the objectives on the performance and connectivity of spiking neural networks. Simulation results on two benchmarks show that Pareto-based evolutionary optimization of spiking neural networks is able to offer a deeper insight into the properties of the spiking neural networks and the problem at hand.

1 Introduction

Spiking neural networks (SNNs) are believed to be biologically more plausible [1, 2] and computationally more powerful than analog neural networks [3]. However, the computational power of SNNs has yet to be demonstrated, mainly due to the fact that an efficient supervised learning algorithm still lacks. In contrast to analog neural networks, for which various sophisticated supervised learning algorithms have been developed [4], only a very limited number of supervised learning algorithms are available for training SNNs, which can be attributed to the discontinuous nature of spiking neurons.

SpikePop [5] is the first supervised learning algorithm that has been developed based on the error backpropagation principle widely used in training analog neural networks. However, SpikeProp has several weaknesses. First, the performance of the learning algorithm is quite sensitive to parameter initialization. If a neuron is silent after initialization, no training is possible for the weights of its incoming connections. As a result, the neuron will never be able to produce any spikes. Second, SpikeProp is only applicable to latency-based coding. Third, SpikeProp works only for SNNs where neurons spike only once in the simulation time. Fourth, SpikeProp has been developed for training the weights only. To address these weaknesses, a few algorithms extending the SpikeProp have been suggested [6–8].

Both SpikeProp and its variants are gradient-based learning algorithms, where simplifications have to be made to apply the gradient method. To resolve this problem inherent to the SpikeProp learning algorithms, evolutionary algorithms [9, 10], which have shown to be very successful in training analog neural networks [11], have also been employed for training spiking neural networks. For example, the connectivity and the sign of the connectivity (the neuron is excitatory if the sign is positive and inhibitory if negative) of a spike response model (SRM) [12] are evolved for vision-based robot control using a genetic algorithm (GA). Similar work has been reported in [13] and [14]. In [13], an adaptive GA is adopted to evolve the weights of the SRM model for robot navigation, while in [14], a parallel deferential evolution has been employed to evolve the weights of the SRM. Both weights and delays of an integrate-and-fire (IAF) model with dynamic synapses [15] and a SRM are evolved using an evolution strategy [16]. It is found that the performance of the SNNs are comparable to that of the analog feedforward neural network on two benchmark problems.

Encouraged by the success of the Pareto-based approach to machine learning [17], this work employs a Pareto-based multi-objective genetic algorithm to evolve the connectivity, weights and delays of the SRM. Different to single objective optimization, Pareto-based multi-objective learning using multi-objective evolutionary algorithms [18] is able to achieve a number of Pareto-optimal solutions rather than one single optimal solution. By analyzing the trade-off between different learning objectives, such as the performance and complexity, we are able to gain a deeper insight into the neural network model as well as the problem to learn [19], by, e.g., identifying interpretable models and models that can generalize on unseen data from the Pareto-optimal solutions.

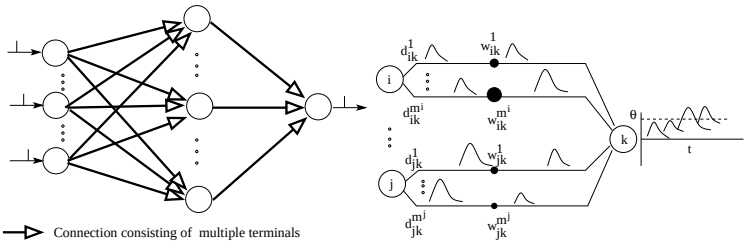
Several objectives concerning learning performance and connectivity can be taken into account in optimizing spiking neural networks for supervised learning such as classification. In addition to minimizing the classification error, we can also minimize the number of connections, or the total length of synaptic connections, which can be replaced by the sum of the delays in the SNN, assuming that the delay between two neurons is proportional to the connection length between them.

2 Spiking Neural Network Model

The spiking network model adopted in this work is the same as the one in [5], which has a feedforward architecture consisting of neurons described by the SRM [12]. The network architecture is shown in Fig. 1(a), where multiple synaptic terminals can exist between two neurons, refer to Fig. 1(b). For clarity, we assume that the network has only one output neuron.

Assume neuron k receives spikes from N pre-synaptic neurons that fire at time instant t_{ik}. For the sake of simplicity, we assume each neuron emit at most one spike during the simulation interval, though this is not a restriction from our evolutionary learning method. The internal state of neuron k at time t, $x_k(t)$, which represents the membrane potential, can be described by

$$x_k(t) = \sum_{i=1}^{N} \sum_{l=1}^{m^i} w_{ik}^l y_i^l(t), \tag{1}$$



(a) (b)

Fig. 1. Feedforward spiking neural networks with multiple synaptic terminals between two neurons (a) The feedforward architecture. (b) Two synaptic connections with multiple terminals.

where m^i is the number of synaptic terminals between neurons i and k, w_{ik}^l is the synaptic strength, and $y_i^l(t)$ is the unweighted contribution of neuron i to neuron k through the l-th synaptic terminal, which can be expressed by

$$y_i^l(t) = \epsilon(t - t_i - d_{ik}^l), \tag{2}$$

where ϵ is a spike response function modeling the post-synaptic potential, t_i is the firing time of neuron i, and d_{ik}^l is the synaptic delay of the l-th synaptic terminal. The spike response function can be described as follows:

$$\epsilon(t) = \begin{cases} \frac{t}{\tau}\, e^{1-\frac{t}{\tau}}, & \text{if } t \geq 0 \\ 0, & \text{if } t < 0 \end{cases}, \tag{3}$$

where τ is the membrane potential decay time constant.

When the membrane potential of neuron k crosses a predefined threshold θ, it will emit a spike. After firing, there is a refractory time during which no spikes can be generated again. As we assumed that at most one spike will be generated during the simulation interval, the refractory time is set to be larger than the simulation interval.

3 A Multi-Objective Genetic Algorithm for Pareto Learning

Evolutionary algorithms (EAs) are well suited for Pareto-based multi-objective learning of spiking neural networks for several reasons. First, EAs do not require explicit gradient information for updating weights, unlike most supervised learning methods. Second, not only weights, but synaptic delays and connectivity of the network such as the number of terminals between two neurons can also be evolved. Third, evolutionary algorithms have shown to be very powerful for multi-objective optimization [18], thus for multi-objective learning.

3.1 Genetic Representation of SNNs

In this work, a genetic algorithm using gray coding has been adopted. Each individual consists of two chromosomes encoding a connection matrix and a weight matrix, respectively. An element in the connection matrix (c_{ik}) is an integer equal to or greater than zero, representing the connectivity from neuron i to neuron k. We investigate both

single-terminal and multi-terminal synaptic connections. In case of single-terminal connections, there is at most one connection between two neurons. There is no connection from neuron i to neuron k if $c_{ik} = 0$. If $c_{ik} > 0$, then it represents the synaptic delay between two neurons. In the multi-terminal case, $c_{ik} > 0$ represents the number of terminals between two neurons, and the synaptic delays of the terminals range from $1toc_{ik}$. The weight matrix (w_{ik}) encodes the strength of the connections between neuron i and neuron j.

3.2 Objectives

Existing supervised learning algorithms for training spiking neural networks minimize the error on training data only. In contrast, a multi-objective learning algorithm can take into account more than one objective with respect to training performance as well as the connectivity of the SNN. In this work, we adopt one of the two error functions, i.e., either the root mean square error (RMSE) or the classification error in percentage, to optimize the performance of the SNN. Besides, either the number of connections or the sum of delays is minimized for optimizing the connectivity of the SNN.

3.3 Genetic Variations and Pareto-based Selection

The uniform crossover and bit-flip mutation are applied at a probability of p_c and p_m, respectively, to evolve the connectivity and weights of the SNN. To select parent individuals for the next generation, we employ the crowded tournament selection method proposed in NSGA-II [20]. First, the offspring and the parent populations are combined. Then, a non-dominated rank and a local crowding distance are assigned to each individual in the combined population. In non-dominated ranking, the non-dominated solutions are found out and assigned a rank of 1. These solutions consist of the first non-dominated front. After that, the non-dominated solutions with rank 1 are removed temporarily from the combined population. Then, non-dominated solutions of the rest individuals in the population are identified, which consist of the second non-dominated front. A rank of 2 is assigned to these solutions. This procedure repeats until all individuals are assigned to a non-dominated front. In the next step, a crowding distance is calculated for each individual with regard to the non-dominated front it belongs to. The crowding distance of a solution is the distance between its two neighboring solutions on the same non-dominated front in the objective space. A large distance is assigned to the two boundary solutions on each non-dominated front. Here, the larger the crowding distance is, the less crowded around the solution.

After non-dominated sorting and calculation of the crowding distance, selection begins. During selection, two solutions are chosen randomly. The solution with the better (lower) rank wins the tournament. If the two solutions have the same rank, the one with the larger crowding distance wins. If two solutions have the same rank and the same crowding distance, choose a winner randomly. This tournament procedure continues until the parent population for the next generation is filled up.

A diagram of the multi-objective genetic algorithm for optimization of the connectivity and weight of SNNs is shown in Fig. 2. The code of our algorithm is developed within the SHARK environment [21].

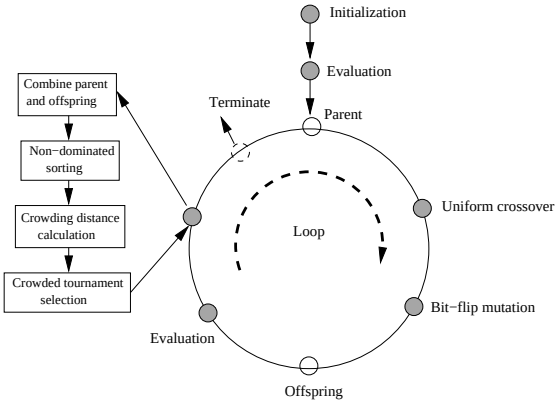


Fig. 2. A multi-objective GA for optimizing the connectivity and weights of SNNs.

4 Empirical Studies

4.1 Experimental setup

The multi-objective learning algorithm has been tested on two benchmark problems from the UCI Machine Learning Repository, namely, the Wisconsin breast cancer diagnosis data set and the Prima Indian diabetes data set. The cancer data consists of 699 instances with 9 input attributes, and the diabetes data contains 768 examples with 8 inputs, all of which are normalized between 0 and 1. The output of both data sets is either 0 or 1, meaning benign (negative) or malignant (positive). The data are divided into training and test sets. Of the cancer data, 525 data pairs are used for training and 174 pairs for test. Of the diabetes data, 576 examples for training and 192 for test.

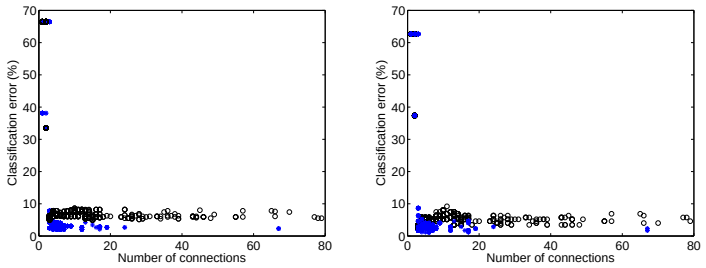
Temporal coding is used for both inputs and output, where the inputs are scaled between 0 ms and 6 ms, and the outputs are normalized between 10 ms and 16 ms for the cancer data, while for the diabetes data, the input is scaled between 0 ms and 16 ms, and the output is set to 20 ms for negative cases and 26 for positive ones. For example, given an input value of 0.1, a spike is generated at time 0.6 ms after the reference time. For setting the reference time, an reference input is included in the network, which serves as the clock, as in [5]. Thus, the number of input neurons equals the number of attributes plus one. For an output value of 0, the output neuron should emit a spike at time 16 ms, while for an output value of 1, the neuron should emit a spike at time 10 ms. The simulation period is set to 50 ms with a step-size of 0.1 ms, the membrane potential decay time constant (τ) is set to 11, and the threshold θ is set to 1000.

Both parent and offspring populations consist of 100 individuals, each of which is composed of two chromosomes, one representing the connectivity and the other the weights. In the single-terminal case, each element of the connection matrix is represented by a 4-bit gray-coded string, which means that the value of the connectivity elements is between 0 and 15. As mentioned in the previous section, a connection value of 0 means that the two neurons are not connected, while a non-zero integer encodes the delay in millisecond between the two neurons. In other words, the maximum delay the genetic algorithm can evolve is 15 ms. In the multi-terminal case, the 4-bit string means that a minimum of zero to a maximum of 15 terminals exist between two neurons. For example, if this value is zero, no terminals exist between the corresponding

two neurons. If this value is three, there are three terminals between the neurons, and the synaptic delay of the three terminals is 1 ms, 2 ms, and 3 ms, respectively. Each synaptic weight is encoded by a 10-bit gray-coded string that is initialized between -10 and 40. The maximum number of hidden neurons is set to 10.

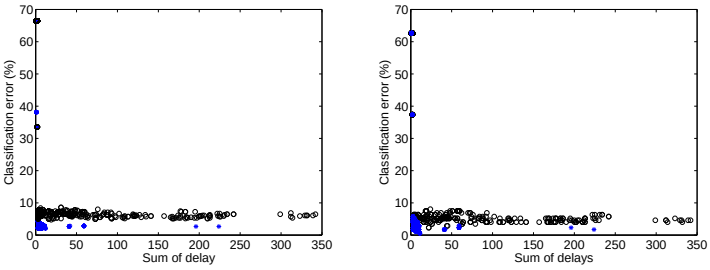
The crossover probability is set to 0.6, and the mutation rate is set to be inversely proportional to the total length of the chromosome. Each evolutionary optimization has been run for 300 generations and the results are averaged over ten independent runs.

4.2 Results from Networks with Single-Terminal Connections



(a) (b)

Fig. 3. Results from 10 independent runs for Case 1 (denoted by asterisks), and Case 2 (denoted by circles) from the cancer data. (a) Training, and (b) test.



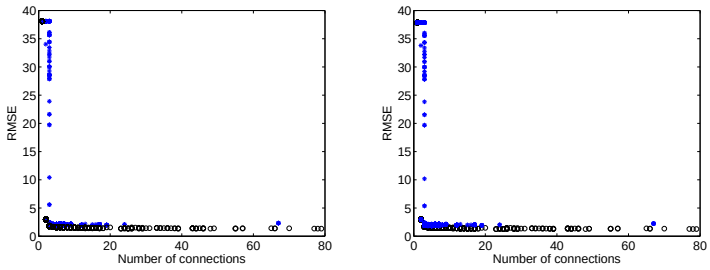
(a) (b)

Fig. 4. Results from 10 independent runs for Case 3 (denoted by asterisks), and Case 4 (denoted by circles) from the cancer data. (a) Training, and (b) test.

We first perform simulation studies for the cancer data when the network has single-terminal connections. We consider four different objective setups, where the two objectives are: Number of connections and classification error (Case 1), number of connections and root mean square error (RMSE) (Case 2), sum of delays and classification error (Case 3), and sum of delays and RMSE (Case 4). The learning results for cases 1) and 2) are presented in Fig. 3(a), where the asterisks denote the results for case 1) and the circles the results for case 2). We can see from the figure that, different to single objective learning, we obtain a number of non-dominated solutions in each run, which

trade the complexity against the performance. Besides, we note that smaller classification errors have been obtained in case 1) than in case 2). This is also true on the test data, refer to Fig. 3(b). In case the RMSE is used as the second objective, the algorithm tends to obtain more complex networks. We also notice that there is a clear knee point on the Pareto front, i.e., the classification error decreases rapidly when the number of connections increases to about 5. After that, improvement in performance is minor as the complexity further increases. Thus, the achievement of the non-dominated front helps us to choose the most economic network for a given problem. Note that in the figures, results from 10 independent runs are merged, thus some of the solutions become dominated by others, though the results from different runs do not vary much. The minimal classification error we obtained on the test data is 1.2%, which is a little better than those reported in [5] and [16], where the test error is 1.8% and 2.4%, respectively. Our results are also comparable to those of analog neural networks with two hidden layers [22], where the mean classification error on the test data is 1.15% and 2.87%, respectively, when direct connections between input and output nodes are present or absent. In [22], the analog neural networks are trained by RPROP, which is a very efficient gradient-based learning algorithm [23].

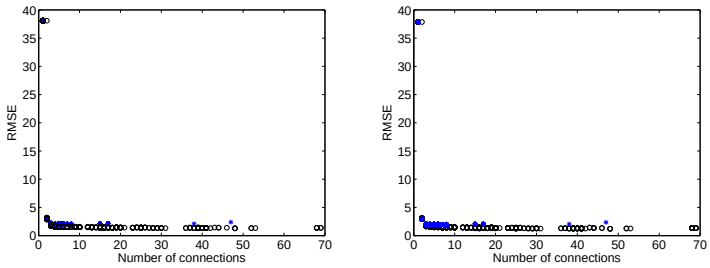
The results for case 3) and case 4) are given in Fig. 4. The performance of the achieved non-dominated NNs is quite similar to cases 1) and 2). To get an impression on the relationship between the two different measures for performance and connectivity, we re-plot the results of Fig. 3 and Fig. 4 in terms of RMSE in Fig. 5 and Fig. 6, respectively. From the figures, the following observations can be made. First, it does not make much difference whether RMSE or classification error is used for optimizing the performance, neither does it, whether the number of connections or the sum of delays is adopted for optimizing the connectivity. Second, in all cases, no overfitting has occurred, which is of great interest compared to serious overfittings in analog networks with high complexity [19].



(a) (b)

Fig. 5. Re-plots of the results from the cancer data for Case 1 (denoted by asterisks), and Case 2 (denoted by circles). (a) Training, and (b) test.

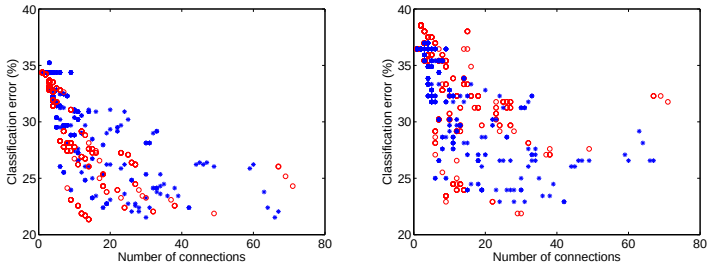
Simulations have also been conducted for the diabetes data. For this data set, we show only the results for Cases 1 and 2, refer to Fig. 7(a) for the training and Fig. 7(b) for the test data. We note that for the diabetes data, the discrepancy between the results from different runs is much larger than that of the cancer data, which suggests that



(a) (b)

Fig. 6. Re-plots of the results from the cancer data for Case 1 (denoted by asterisks), and Case 2 (denoted by circles). (a) Training, and (b) test.

the diabetes data is more difficult to classify than the cancer data. Since results on the diabetes data using SNNs have not been reported elsewhere, we compare our results with those in [22]. In that work, the mean classification error over 60 runs on the test data is 25.83% when an analog network with two hidden layers are used. Thus, the performance of the SNN is better than that of the analog neural network in some of the runs are. Finally, similar to the cancer data, no serious overfitting has been observed for the diabetes data as the complexity of the network increases.



(a) (b)

Fig. 7. Results from 10 independent runs from the diabetes data for Case 1 (denoted by asterisks), and Case 2 (denoted by circles). (a) Training, and (b) test.

4.3 Results from Networks with Multi-terminal Connections

We now discuss briefly the results on SNNs with multiple terminals between two connections. The results for the cancer data are presented in Fig. 8, where the number of connections and the classification error are minimized. Since there can be multiple terminals between two neurons, the number of connections becomes much smaller than the single terminal cases, though the performance on the training and test data is similar.

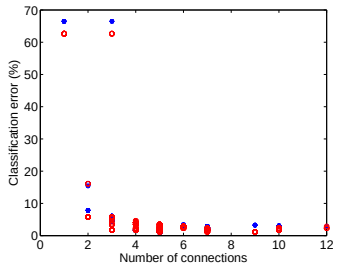


Fig. 8. Results on the cancer data from SNNs with multiple terminals. Asterisks denote the training data and circles the test data.

5 Conclusion

In this paper, we suggest a method for optimizing the performance and connectivity of SNNs using a Pareto-based genetic algorithm. Compared to existing supervised learning algorithms for SNNs, the Pareto-based approach considers both learning performance and connectivity of the SNNs without aggregating the two objectives. Either the RMSE or classification error has been minimized for optimizing the performance, and either the number of connections or the sum of delays has been minimized for optimizing connectivity.

One main merit to use the Pareto-based learning is that we can obtain a number of Pareto-optimal solutions that trade off performance against complexity. By exploiting the tradeoff solutions, we are able to gain an insight into the learning properties of SNNs, e.g., the minimal complexity needed for a given problem. Besides, we are able to investigate whether overfitting occurs when the complexity of the network increases. In our study, no overfitting has been observed, no matter how high the complexity of the network is. This is a very interesting finding, however, further experiments must be performed to confirm this observation.

No conclusion can be made on whether the RMSE or classification error should be used for classification. We have also shown that complexity of the SNNs are comparable, when the number of connections or the sum of delays is used to optimize connectivity.

One of the future work is to compare SNNs using other coding schemes than temporal coding and to consider SNNs that can emit multiple spikes within the simulation time. In addition, studying the relationship between functionality and connectivity of large scale spiking networks using sophisticated network simulation software like NEST [24] is our further research target.

References

1. S.J. Thorpe, A. Delorme, and R. Van Rullen. Spike-based strategies for rapid processing. *Neural Networks*, 14(6–7):715–726, 2001.
2. S.M. Bohte. The evidence for neural information processing with precise spike-timing: A survey. *Natural Computing*, 3(2):195–206, 2005.

3. W. Maass. Networks of spiking neurons: The throd generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.
4. R.D. Reed and R.J. Marks II. *Neural Smithing - Supervised Leanring in Feedforward Artificial Neural Networks*. The MIT Press, 1999.
5. S.M. Bohte. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48:17–37, 2002.
6. O. Booij and H. Nguyen. A gradient decent rule for spiking neurons emiiting multiple spikes. *Information Processing Letters*, 95(6):552–558, 2005.
7. B. Schrauwen and J. Van Campenhout. Extending SpikeProp. In *IJCNN*, pages 471–476, 2004.
8. B. Schrauwen and J. Van Campenhout. Backpropagation for population-temporal coded spiking neural networks. In *IJCNN*, pages 3463–3470, 2006.
9. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
10. H.-P. Schwefel. *Evolution and Optimum Search*. John Wiley, 1994.
11. X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.
12. W. Gerstner. Time structure of the activity in neural networks models. *Physical Review*, 51(1):738–758, 1995.
13. H. Hagras et al. Evolving spiking neural network controllers for autonoumous robots. In *IEEE Intternational Conference on Robotics and Automation*, volume 5, pages 4620–4626, 2004.
14. N.G. Pavlidis et al. Spiking neural network training using evolutionary algorithms. In *IJCNN*, pages 2190–2194, 2005.
15. M. Tsodyks, K. Pawelzik, and H. Markram. Neural networks with dynamic synapses. *Neural Computation*, 10:821–835, 1998.
16. A. Belatreche, L.P. Maguire, and M. McGinnity. Advances in design and application of spiking neural networks. *Soft Computing*, 11:239–248, 2007.
17. Y. Jin, editor. *Multi-Objective Machine Learning*. Springer, 2006.
18. K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, 2001.
19. Y. Jin and B. Sendhoff. Pareto-based multi-objective machine learning: An overview and case studies. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2007. accepted.
20. K. Deb, S. Agrawal, A Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature*, volume VI, pages 849–858, 2000.
21. http://shark-project.sourceforge.net/.
22. L. Prechelt. Proben1 - A set of neural network benchmark problems and benchmark rules. Technical report, Falkutät für Informatik, Universität Karlsruhe, 1994.
23. M. Riedmiller and H. Braun. A direct adaptive method for faster backpropgation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, volume 1, pages 586–591, 1993.
24. http://www.nest-initiative.org/.