ijar
ISSN NO. 2320-5407

Journal Homepage: -www.journalijar.com
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/1627
DOI URL: http://dx.doi.org/10.21474/IJAR01/1627

RESEARCH ARTICLE

A REVIEW OF OPTIMIZATION TECHNIQUES IN ARTIFICIAL NETWORKS.

Omid Ghasemalizadeh, Seyedmeysam Khaleghian and Saied Taheri.

……………………………………………………………………………………………………....

Manuscript Info

Manuscript History

Received: 12 July 2016
Final Accepted: 13 August 2016
Published: September 2016

Key words:-
artificial networks, optimization techniques, parity, alphabetic fonts, Monk problem

Abstract

Previous studies showthat neural networks are implemented on various problemssuch as regression and classification successfully. Optimization process of a neural network is highly in accordance with unconstrained optimization theory and many efforts have been done to accelerate this process. Particularly, various algorithms have been developed by numerical optimization theory to speed up neural network optimization. Additionally, conventional innovative approaches like variable learning rate or momentum lead to a considerable improvement in this regard. In this paper, first, we introduced optimization techniques in neural networks and then, compared the methods in three problems of parity-n, alphabetic fonts, and Monk. We attempted to conduct a study at a large scale on performance of the presented algorithms and identify their probable advantages.

……………………………………………………………………………………………………....

Introduction:-

There are many optimization techniques which are appropriate to optimize neural cost functions and neural architectures. We hope that the present article can pave the way for further researches in this regard although, global optimization is not the only solution for local minima problem.

One of the simplest and most commonly used techniques is the method based on momentum terms added to gradient. This method has been widely discussed in all books related to neural networks [1, 2, 3, 4, 5, and 6]. Although using momentum does not lead to a global optimum, it can be helpful to avoid weak local solutions. Maybe the most obvious and less common method to find optimal solution is based on initialization which is followed by gradient optimization [7, 8 and 9]. Initialization method should make adaptive parameters close to global optimum.

Accidental initialization or biases and small weights which are usually used for neural networks may not be adequate in case of large optimal parameters. In a series of experiments, Hochreiter and Schmidhuber [10] observed that high number of iterations in accidental initialization (guessing weights) leads to faster convergence compared to using complex versions in gradient method.

Gradient optimization methods usually cannot compensate bad initials for weights and biases and get caught in local optima. A good strategy might be to use accidental weights as soon as optimization process slows down. Using a good initialization or a non-linear global optimization method may be a better strategy.

Corresponding Author:-Omid Ghasemalizadeh.

Direct method of finding optimum structures is defined as to use "educated guess" for good structures and then, select structure with higher probability. Some optimization methods such as genetic algorithm [11] or simulated annealing [12] are employed to produce new data architectures through high-quality previous data estimation (fitness in genetic terminology). This strategy is based on the assumption that data - which is measured by error in optimization in validation set- is a in form of a smooth function of adaptive parameters or data topology.

Adding a neuron or omitting a relation and subsequently, re-optimizing data should not have a significant effect on data quality. If objective function is chaotic, all optimization methods are hindered finding global optimum. However, since the performance of many neural architectures are investigated and the best one is selected, such strategy may create data with higher quality compared to given data quality. Investigating hypotheses about the smoothness of data and describing characteristics of the space of all possible architectures, whether theoretically or via simulation, shall be of interest to researchers.

Maybe it is a probable accurate prediction. Although optimization can solve some problems which are beyond back propagation algorithm, the best error minimization method cannot compensate the defects of selected data or mathematical model.

For example, assume classification limits are that are clearly presented by two hyperplanes with a simple logical rule $((x_1 > 1) \wedge (x_2 > 1))$ but, there is no way to precisely present them using the sum of two soft sigmoidal functions in Multilayer Perceptron (MLP) data. Increasing sigmoidal functions ranges to present decision borders in vicinity of point (1,1) leads to issues with back propagation optimization or gradient-based methods. It is due to the fact that the volume of input space in which sigmoids are changing (and gradients are non-zero) is rapidly decreasing.

In limit sigmoidals, functions are changed into step-functions but gradient techniques like back propagation cannot be employed to create such transition. Accordingly, for some of optimization sets, no changes will lead to accuracy improvement in gradient-based optimization law or data architecture. Most of total optimization techniques have not such disadvantages and can optimize slopes as well as other adaptive parameters without facing numerical instabilities.

A good example in real world is hypothyroid dataset [13] and its best optimized MLPs have still an error about 1.5% [14];while, logical rules [15] decrease this error down to 0.64% (since 3428 cases were provided for testing, this improvement is considerable). Most of researches focus on neural networks on data architectures and optimization rules but selecting neural selection functions may have a significant effect on the complexity and performance of neural models performance [16].

Selecting optimization technique can lead to a considerable improvement in data quality as well as improvement in convergence speed of optimization algorithm. Substituting gradient-based back propagation algorithms [18 and 19], using smaller and more compressed neural networks, with global optimization [17] can makes much complicated problems feasible.

For instance, using a new mixed method of local gradient method and search space global exploration,Wah and Shang [20] found appropriate solutions for two-spiral benchmark problem only through 4-6 latent neurons. Previously, the smallest MLPs (which were constructed using cascade correlation algorithm [21]) required 9 neurons and 75 weights; optimization process was also very sensitive to the primary conditions. According to Wah and Shang [20], the chance of finding a good local optimum using a large data set is higher compared to using a small one. It is due to the fact that small data is strongly uneven. Therefore, it is easier to achieve good results using one of gradient-based methods in larger data.

Total optimization should specifically be useful for smaller data. Using global methods should also improve total rules which have been extracted through neural networks [22].

Few optimization techniques have been employed in neural networks so far. Most of techniques have been only limited to research articles regarding physical, chemical, engineering, or financial problems. In present work, we have attempted to deeply investigate these global optimization techniques.

Investigating Optimization Techniques:-

Monte Carlo

In the simplest method of Monte Carlo (MC), a new vector of P parameters is randomly produced by changing only one single parameter or a group of parameters. This change for optimizing neural data structures may include adding or omitting a relation or a neuron with several relations and might be followed by gradient-based optimization.

By using gradient-based optimization regarding MC optimization, obtaining global optima is not guaranteed; although, it is relatively fast. After producing and optimizing a certain number of data, the best ones are selected and employed in cross-validation test. This method is not as complex as Genetic Algorithm and its implementation is easier. Another technique is to use a separate set of discontinuous parameters of P_a and continuous parameters of P_w (i.e. weights and biases) and both are optimized using Monte Carlo method. In this case, selecting architecture is followed by slower search for global optimization of cost function. After adequate yet long time, producing parameters randomly leads to exploring global parameter space but, if the number of parameters is high, computations time may be prohibiting.

Although in MC parameters are randomly selected, they may be extracted from probability distribution and total constraints may be easily implemented in favor of small data. Another technique is to use quantized weights and biases during MC search which is followed by short gradient-based optimization (without quantization). An interesting development in MC method was proposed by Dittes [23] regarding N interacting spins but this method has a general use. Energy or optimized error is a set of a spin conditions. Additionally, interacting conditions of two spins are supposed to be interacting conditions of K^2- spin.

A total set of energy functions with the assumption of K=1…N is defined as spin energy. Global Optima was searched for all these energy perspectives: probability distribution was defined for parameter changes and selecting one of the perspectives. Energy for each of these perspectives should be appropriate for total energy but local minima should be in another place. The superiority of this method, particularly for difficult optimization problems was shown compared to other MC methods with its application in several spin glass and traveling salesman problem. Regarding neural systems, the proposed method sits somewhere between online optimization method and batch training. In online methods, parameters are set after displaying each new vector; however, in batch training, parameters are changed at the end of a period in which all optimizations have been presented.

This method can be applied along with every other method which determines adaptive parameters change such as genetic algorithms or simulated annealing. Besides error value, this method attempts to use configurations of various training data vectors as well. This method has not been tested in neural networks so far.

Simulated Annealing and Its Variants:-

This method was introduced by Kirkpatrick et al. [12] in 1983. Simulated Annealing (SA) has been inspired from annealing crystals which reach to the lowest level of energy, the complete crystals which are very slowly annealed. This method is practically very difficult; even very good crystals may have some defects, which indicate the difficulty of total optimization process such that even nature faces problems in this regard. High temperature allows atomic configurations to reach a higher state of energy and overcome energy borders and prevents internal stresses due to defects. Simulated annealing has been used in many applications in all branches of science and technology. In fact, simply, simulated annealing method adds an important sampling in accordance to Boltzmann distribution (in thermodynamic) to select new parameter vectors. This fact causes to complete adaptive parameters vector of $P = (P_1,\ldots,P_{n)}$from the initial value to optimum value of error function. Figure below shows a one-dimensional cost function optimization by SA.

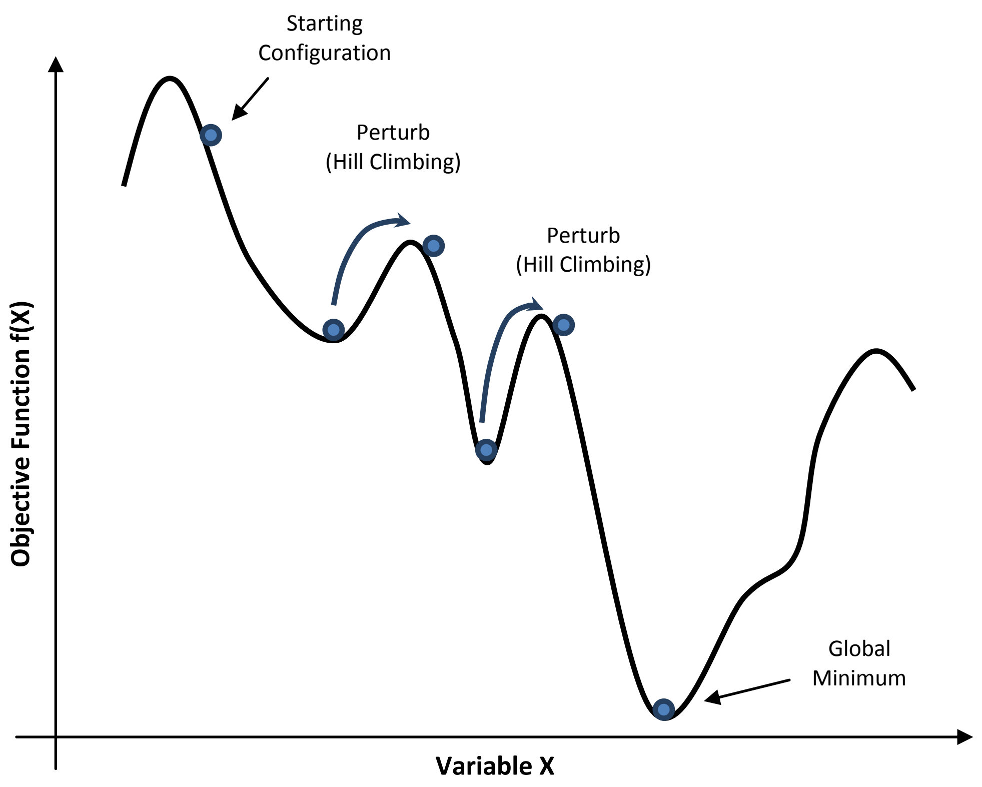

Figure 1:- Simulated Annealing optimization of a one-dimensional objective function

Mean-field Annealing (MFA) is a well-known approximation for stochastic search which is obtained for quadratic errorfunctions of quenched variables. However, MFA methods are easily used in problems such as Hopfield to optimize their energy; they also show good but unexplained performance for quadratic error function or quasi-linear functions which is used for MLP or other feed-forward networks data.

Adaptive Simulated Annealing:-

Adaptive simulated annealing (ASA), previously called very fast simulated reannealing (VFSR) [25] employs a different probability distribution for various parameters [24]. SA is not popular among neural data researchers who work on MLPs. Except than the interesting article of Engle [27] in which adaptive parameters are separated and presented, harmonic data and Boltzmann machines are based on simulated annealing method [28 and 29].

SA with Gibbs sampling technique is used in Bayesian methods for neural networks [30]. In a study [31], it was revealed that MFA in neural networks outperforms optimization techniques. SA method is employed for solving inherent optimization problems in vector quantization methods [33], feature weighting in (learning vector quantization) LVQ data [34], probabilistic networks optimization [36], and recursive neural networks with chaotic behavior [37].SA can be also combined with Dittes method which has not been yet implemented. SA, in most of optimization problems, has showed better performance compared to other methods [24 and 36]. There is a need of further investigation regarding ASA application in neural networks.

Alopex:-

In Alopex algorithm, a certain type of simulated annealing is used [39]. Alopex algorithm is based on various simple ideas which compete with back propagation algorithm.

Alopex algorithm has been tested on few issues so far and has resulted in promising manner. For example, it has been used to solve parity problems as well as all standard machine learning benchmarks: it solves three problems of Monk [40] with the accuracy of 100% (except than MLP2LN method [22 and 41], it is the only data which has been able to fulfill this work); however, no report of noisy data results in the real world has been presented.

Reactive Tabu Search:-
Reactive tabu search is based on a very simple idea [42 and 43]. This search starts with a random point and the best primary movement is selected. Maintaining search path and preventing system from revisiting identical areas, cycles are avoided. Reactivetabu search was used on many combinational optimization problems which obtained promising results. To distinguish interesting events in optimization, high energy physics was used [24] and the best results were obtained for presenting one-bit weights (which was interpreted as weight values of $W_i=\pm5$). Generalization levels reached to 90% while in MLP, standard has only reached to 62%.
Unfortunately, the researcher had no effort to embed zero weight values. Results show that in a large data, search space exploration may be more important than finding precise values for weights.

MultisimplexMethods:-
Linear least square SIMplex (LLSSIM) is another interesting method in global optimization which is based on multisimplex optimization. This method has been recently proposed by Gupta et al. [45]. They reported remarkable results of using this method in neural networks.

Multisimplex method is used for global optimization similar to multi-level-single-linkage (MLSL) stochastic methods. This method is a specific type of clustering method [46]. Here, a cluster is defined as a set ofpoints corresponding to a domain which includes only one optimum. Single-linkage methods evaluate functions over a set of sample points, find the best solution, impose local optimum, and create a cluster by adding some points in vicinity of the optimum. Some points of the primary sample are placed in the cluster (i.e. local optimization is placed in the cluster’s optimum) but the rest of them will be at a distance far from the critical point relative to the cluster (i.e. relative to the nearest point in the cluster). The farthest points should form a new cluster.

Therefore, space is divided into two clusters or two basins of attractors in local optimal gradient dynamics. The ample set is constantly expanding; so, even finite sampling method within each cluster will be also obtained as a result of all optimums.

In MLSL method, local optimization is applied in all points which were sampled at the beginning unless they are closer than the critical distance. This method has not been used in neural networks so far.

Global-Local Hybrid Optimization Methods:-
Baba [47] and Baba et al. [48] described the first hybrid algorithm for total optimization in neural networks. Conjugate gradient method with linear search is employed to find a local optima. It is changes into a global mode when error reduction reaches below a threshold to escape the local optima. When error reduction is not significant, local optimization is re-triggered. Baba [47] employed random optimization method of Solis and Wets. This method guarantees convergence to global optimum.

This algorithm converges towards global optimum by utilizing probability. Application of local optimization in neural networks was the first method which used SCG and started from weights in ±0.77 interval. Random lines always start from origin (in RLS algorithm, these lines are generated from W_i). One of the most effective one-dimensional search techniques is called P^* [15].This technique uses a complex strategy and creates a statistical model (Weiner process) to estimate position of local optima as well as a quadratic estimation to find the final value. The length of the line is scaled with a factor appropriate with Gaussian distributed variable; while, components are randomly and uniformly selected in a specific interval. Therefore, the most of generated lines are relatively short.

The cost of global strategy, compared to the cost of local searches, is relatively low in P^* method (usually about 10%). The most important parameter which may be changed by user is weights interval (in some experiments, even ±10 is very low and it depends on preprocessing of the input data).

Smoothing Algorithms:-
Smoothing algorithms are designed and used for optimal search of potential energy functions [58]. The idea behind these algorithms is to smooth out the objective function into a simpler function. By deforming the potential energy using the smoothing procedure, the potential wells evolve, they become shallower, and their positions and shapes gradually change. Eventually, one ends up with one well that corresponds to the global minima. Therefore, the global minima of the evolved hypersurface can be found using any local optimization methods. As said before,

position of newly formed global minima might have been changed. This issue is resolved by proposing a revising procedure.

This method has been employed in some of physics-chemistry problems [58]; however, it has not been used in neural networks so far. The ability of computing high-order derivatives is systematically needed for direct use of exponential operator. Diffusion equation can be solved in specific cases, for example, in error functions of polynomial (the mix of polynomial factors multiplied by exponential factors) which is not related with neural networks approximately. Other methods which use optimization in physics-chemistry should be investigated as well [59. 60 and 61].

Branch and Bound Methods:-
Branch and bound methods present lower borders in target function. They are similar to discrete minimization methods which are used in combinatorial AI searches [62]. The history of branch and bound methods (B & B) is explained in [63]. One of the advantages of using these methods is that they need no information about objective function and can be combined with many exploratory techniques to discover search space. These methods may determine error level and obtain local optimal and saddle points in vicinity of global optima which can be useful in some applications.

This problem is called root problem across the domain. To compute upper and lower bounds, a method should be defined and if both borders are in accordance to current problem, an optimum has been determined; unless, domain is divided into smaller sub-domains and borders are investigated for them. A graph with search nodes is defined and expanded recurrently. Finding a local optimum in some of sub-domains makes it possible to omit pruned tree and all nodes with lower border found at the top of local optimum.

Implementation of these methods for neural optimization has not been clearly expressed in literature.

Interval-Based Methods:-
Interval-based method [64] in which information about objective function in a box, cone, or simplex-shaped areas is available, is another interesting topic that can be employed in combination with branch and bound (B&B) methods. Interval computations are getting more popular day by day. They have been also developed as toolboxes and extensions of Fortran 90, Maple and Mathematics. Interval methods which are used for global optimization are occasionally faster than point methods.

List of boxes includes local and total optimums. Hansen [64] modified this algorithm and added various techniques to it to omit some parts of boxes. Recently formulated Back-Boxing method seems to be the most effective interval computations and rectangular parts-based partitioning. This method is used with constrained damped Newton's method for real local optimization. Using this local method is necessary since branch and bound algorithm spends most of its time to local optimums. Newton's method needs to compute Hessian interval. However, applying it in neural networks is relatively costly.

Back-Boxing is the process of identifying a box where error function is convex. Finding the largest box is a non-trivial issue. Boxes around inflection points are avoided. In this algorithm, there are three types of boxes. First, boxes with uncertain content; second, boxes minimized to the smallest size possible and contained optimums, and third, a list of convex boxes with convex error function. The algorithm itself is fairly complex and has been explained in details in [63].Global optimization methods have not been employed in neural networks by researchers; however, their implementation on Neural Networks is not much complicated. Interval computations have been reported to be used for missing values in classification and regression [65] but the author used only local optimization methods.

Genetic Algorithms:-
It seems that the high popularity of Genetic Algorithms (GA) in neural data optimization is originated in biological motivations of these two methods although, in practical applications, it does not matter whether there is a biological motivation or not. Genomes mutations for acceptable local optima is similar to random stages in Monte Carlo algorithm. Therefore, it seems that "survival of the fittest" can play a fruitful role, especially that crossovers increase the probability of leaving local optima. A schematic diagram of GA is depicted in

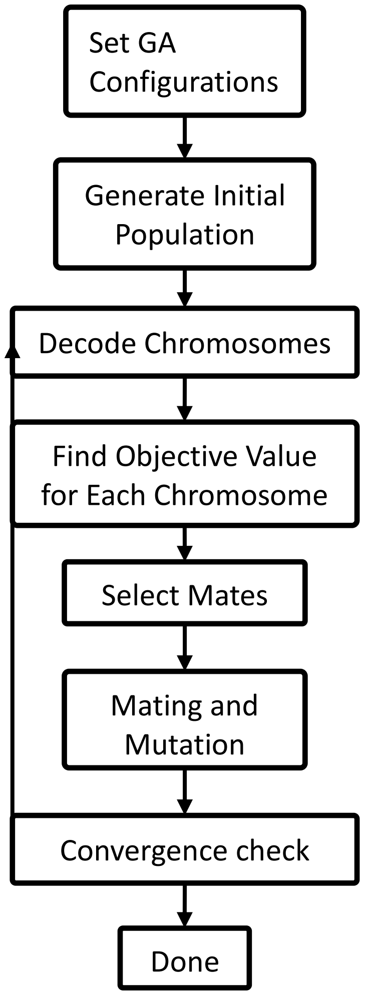

Figure 2:- GA procedure flowchart.

However, unlike simulated annealing method and some other Global Optimization methods, GA methods do not guarantee global convergence. Therefore, the results obtained from genetic algorithms should be compared with other Global Optimization methods, since its performance is not guaranteed with Neural Networks.

However, success of Genetic Algorithms depends on precise analysis of the problem. One of the main applications of GA is when problem is defined on a large space [11, 66, 67, and 68]. GA is inspired from adaptive and optimization capabilities of natural species.

Compared to other methods, GA has a remarkable ability to explore discontinuous spaces in solutions (which is common in most of global optimization methods but gradient-based methods lack this capability). Also, GA has knowledge of optimal points history and domain-based information.GA techniques explore in parameters space which is guided by a fitness function and look for other optimal points in parallel and do not focus on only one local minima. MLP data is the most common among various kinds of data model. They are common due to their proven ease of use [70, 71 and 72]. The efficiency of optimization and the quality of generalization are related with a strong relation with neural data topology.

The number of neurons, their organization in layers as well as their connection scheme will have a significant effect on data optimization and their capacity for generalization [73]- [76]. Using inappropriate data architecture influences the performance of various factors such as optimization time, convergence, and generalization capacity. Genetic connectionism based on evolution and optimization in a system is performed through combining connectionist methods and genetic search techniques. Evolving neural networks have been employed in some research projects [69, 73]-[88].

Based on level and method of integrating connectionist and GA methods, these methods can be divided into four groups. The first group consists of methods which use GA for preprocessing of Global Optimization [89, 90]. The second group uses GA for neural network optimization which typically consists of weights optimization in a neural data with predefined topology [81, 83, 91]-[94]. The third group uses GA to select typology of neural data [75, 79, 84, and 95]-[98]. Finally, the fourth group is a combination of the three mentioned groups [99 and 100].

Supportive Methods:-
In supportive methods, GA(s) are used to help neural networks and vice versa, i.e. neural networks are used to optimize GA. There are some instances in the following:

Neural Networks Help to Genetic Algorithms:-
XROUTalgorithm was developed by Kadaba et al. [117 and 118] to remove vehicle routing problem. This problem includes optimizing the distance driven by a vehicle with allocating stopping points and servicing orders
To select the optimal route, parameters were simulated by chromosomes. Neural networks were used to generate primary population in two searches which were performed by GA.

Genetic Algorithms Help to Neural Networks:-
Genetic algorithms can be used to support Neural Network researches:
1. Selecting input optimizers or changing feature space
2. Selecting an optimization law for data and its parameters
3. Analyzing a neural data [90 and 119].

Optimization preprocessing:- Kelly and Davis [120] employed GA to find rotations of data vectors and scaling factors for each attribute which cause to improve classifier performance. Other methods focus on reducing optimization space. Chang and Lippmann [121] also used GA to reduce space of data. This algorithm builds a set of new input optimization from the main set; for example, polynomial functions generate new features of raw optimization. Drabe et al. [122] employed GA to cluster subtasks of a complex task which had to be learnt by neural data. This research is a rare study which attempts to employ neural networks in optimization-related complex problems. GA(s) are often used to select features before MLPS optimization [123], Kohenen data, and/or quantization vector algorithms[see 87].

Modifying optimization rules and parameters:-Belew et al. [96] used GA to determine learning coefficient and momentum for MLP data optimization. Convergence speed was improved compared to hand-assigned values. Harp et al. [79] employed GA to dynamically modify learning coefficients based on the number of epochs. Schafier et al. [85] compared learning coefficient, momentum and connection weights. Chalmers [78] encoded optimization law in a chromosome and changed this law by observing the performance of previous epochs. GA has been also employed for initializing radial basis networks [124 and 125] and optimizing cellular neural networks [126].

Neural data analysis using GA:-Optiz and Shavlik [127] used GA to describe the behavior of neural networks through defining a function relating input and output data.

Collaborative Methods:-
The idea of optimizing data using GA during optimization process is very popular [79, 83, 85, and 128]. Genetic Algorithm techniques were used to determine connection weights by mean-square quadratic error function as an adaptive function. However, it was a time-consuming process; even if some exploratory methods had been used to reduce computation time. Other total optimization methods in neural networks are exclusively used for error function optimization. A more natural method to combine genetic algorithm and neural networks is to use genetic search methods to find an optimum data topology.

Hybrid Methods:-
Methods that use GA to modify neural data connections weight, compared to back propagation-based methods, have less efficiency and higher computation cost. Another probable use of GA is initializing adaptive parameters as well as determining learning coefficient and momentum. In some applications of neural data where using back propagation is not possible, e.g. in recurrent data, GA presents an alternative solution for gradient optimization algorithms [129].

Applying GA has resulted in promising results in MLP data topology optimization [80, 93, 104, 121, and 130], RBF data parameters optimization and Kohonen data structures [131 and 132].The characteristic of a mapping function which encodes data in a strain of genomes has been recognized as "a chromosome" and does not such optimization. Some encoding methods of MLP data have been used in chromosome [84, 86, 103, 133, and 134] and some of them will be discussed in the following.
Defining an adaptive function that considers various aspects of a hybrid optimization system is another important problem. Before discussing on this subject, a process of genetic search process in an optimal neural data is described.

Particles Swarm Optimization (PSO):-
PSO [149 and 150] has been inspired from evolutional computations and artificial life. It is highly different from Genetic Algorithm in a sense that each particle stores its best fitness value throughout the optimization process. Swarm refers to interactive points called "particles" that flow in a space of adaptive parameters. Each particle with a low velocity tries to optimize target function through a collaborative search processes.

Also, it has not been proven that this method, compared to GA or even standard back propagation method, has the ability of convergence but is has shown higher velocity and ability to find better solutions [150].

Methodology:-
Heuristic parameters were considered$\eta_0 = 0.1, \sigma_1 = 10^{-4}, \sigma_2 = 0.5, m = 0.9$for all algorithms and$3 \leq M^k < 15$for all experiments which give us the best general results. This implementation was performed on a computer processor (3.2MHz, 2Gbyte RAM) using MATLAB Neural Network Toolbox version.

For each tested problem, a figure is presented that summarizes the performance of investigated algorithms. The employed parameters are as following:

MIN: the minimum number of epochs
MAX: the maximum number of epochs
MEAN: mean value of epochs
S.D.: standard deviation
Succ: simulation went beyond 100 experiments in predetermined error domain

If an algorithm cannot converge in epoch limit, its failure is considered in neural network optimization but its results are not considered in analysis of the algorithm.

Results:-
N-Bit Parity Problem:-
N-bit parity problem can be considered as a generalization of exclusive-OR (XOR) problem. N-bit parity problem optimizes a Neural Network to produce set of 2 modes. This problem which is usually used to evaluate the performance of an algorithm, has been a well-known benchmarking problem and recognized as a challenging problem for finding local minima.

Here, two types of above mentioned problem, parity-2 and parity-5, are considered for this chapter benchmark test. All networks are based on hidden neurons in hyperbolic tangent activations and one linear output neuron. For parity-2, neural networks with two hidden nodes and for parity-5, neural networks with seven hidden nodes were employed. Stopping criterion at the limit of 1000 and 2000 rounds for each sample was placed in error$\leq 10^{-2}$.

Figure 1 shows the average performance of algorithms presented for parity-2 problem. Generally, quasi-Newton algorithms have shown the best performance since the least average value of epoch has been reported for convergence. It indicates highest rate of success in optimizing a feed forward neural network. Among these algorithms, B&B, compared to two other quasi-Newton algorithms, shows better performance in optimization.

Regarding conjugate gradient algorithms, PSO shows the probability of successful optimization, demonstrating the lowest average number of epoch unlike other conjugate gradient algorithms. Additionally, it should be noted that PSO shows the highest probability (69%) in successful optimization of a recurrent neural.

Regarding the first-order algorithms, back propagation algorithms of B&B and PSO require least amount of computation in average epoch value and their results are also similar to those of the second-order algorithms.

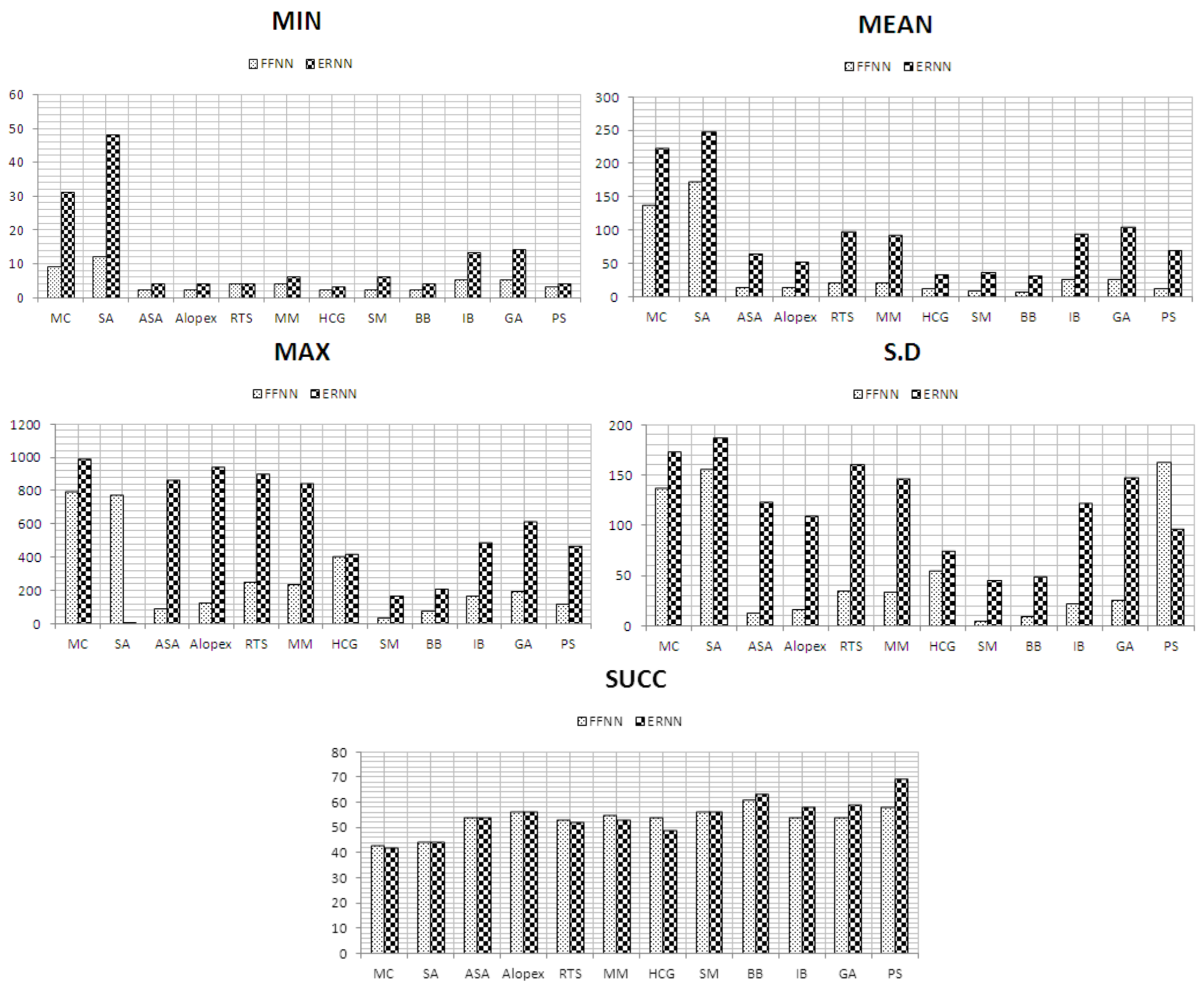

Figure :-. N-parity comparison.

Alphabetic Font Learning Problem:-
The problem recognizing letters is one of the oldest problems in evaluating the performance of a neural network learning algorithm. In terms of binary values, each letter has been defined on a 5×7 grid. Each network is based on 30 hidden neurons in terms of logarithmic structures and 26 neurons in terms of linear output. When error$\leq 10^{-1}$in predetermined limit of 2000 epochs, optimization is considered successful. However, due to big size of the network, quasi-Newton algorithms could not be applied for this specific topology.

Figure 2 shows a comparison of the performance of algorithms presented in alphabet font problem. All algorithms show a high probability (100%) of successful optimization in both architectures of the network. Therefore, computational cost is probably the most appropriate index for measuring the efficiency of algorithms. The performance of PSO, in terms of average number of epochs, is better than other algorithms. Especially, the required average number of epoch is seven times less than others unlike the classic conjugate gradient algorithm that show 84% improvement at order. Therefore, PSO provides faster and more stable optimization (Figure 2). Regarding the first-order algorithms, back propagation algorithms of B&B and PSO, compared to other first-order algorithms, show less (34% to 97%) average number of epochs.

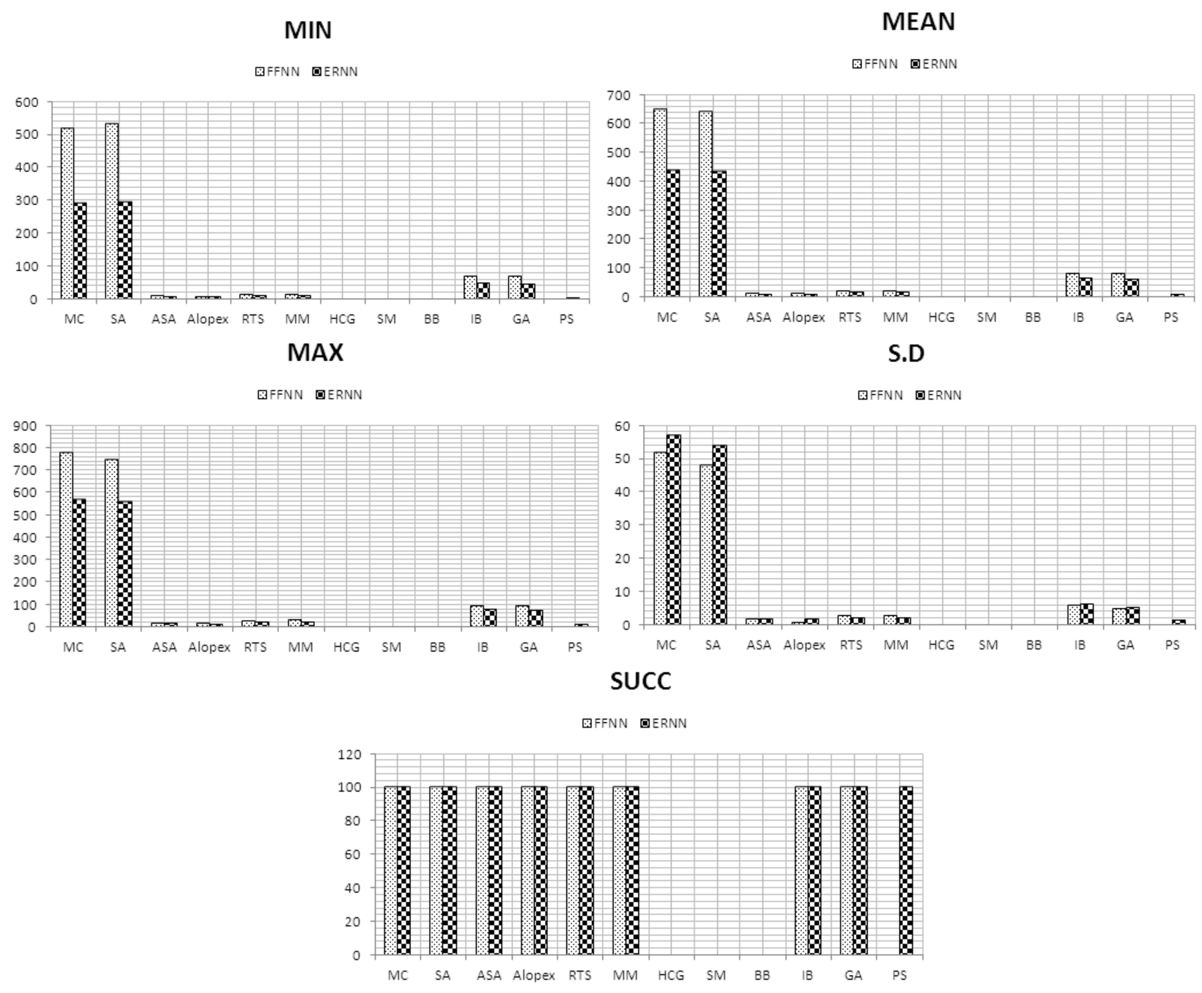

Figure 2:- Alphabetic font learning problem.

Monk Problem:-

Monk problem is a collection of three binary classification problems relying on robot domain in which robots are described with various attributes. The problem of Monk has been presented with three variable models:

1. Monk-1: includes 124 schemes which were randomly selected from dataset for optimization and 308 remaining schemes were used for generalization testing.
2. Monk-2: consists of 169 species which were randomly selected from dataset for optimization and the rest were used for testing.
3. Monk-3: includes 122 schemes for optimization and the rest were used for testing.

In simulations, neurons of hidden layer identical with the found number were employed. The figure presents simulation results for Monk-1 problem. In both architectures of network, back propagation and momentum could not converge at predetermined error limit. Quasi-Newton algorithms show the best performance since it has a significant rate of success (100%) and requires the least average number of epochs. Notably, B&B has a better performance than SM.

In general, conjugate gradient algorithms showed the worst performance among the second-order algorithms although the performance of PSO was better than other conjugate gradient algorithms (shows approximately 70.8% to 76.5% less average number).

Regarding back propagation variable models, it should be reminded that variable learning rate has a significant effect on the improvement of back propagation efficiency, especially adoption with B&B and PSO stages [14]. Additionally, it should be noted that using non-monotone strategy has a considerable effect on all algorithms. Figure 3 compares the performance of ASA and ALOPEX and the performance of HCG and B&B considering number of epochs.

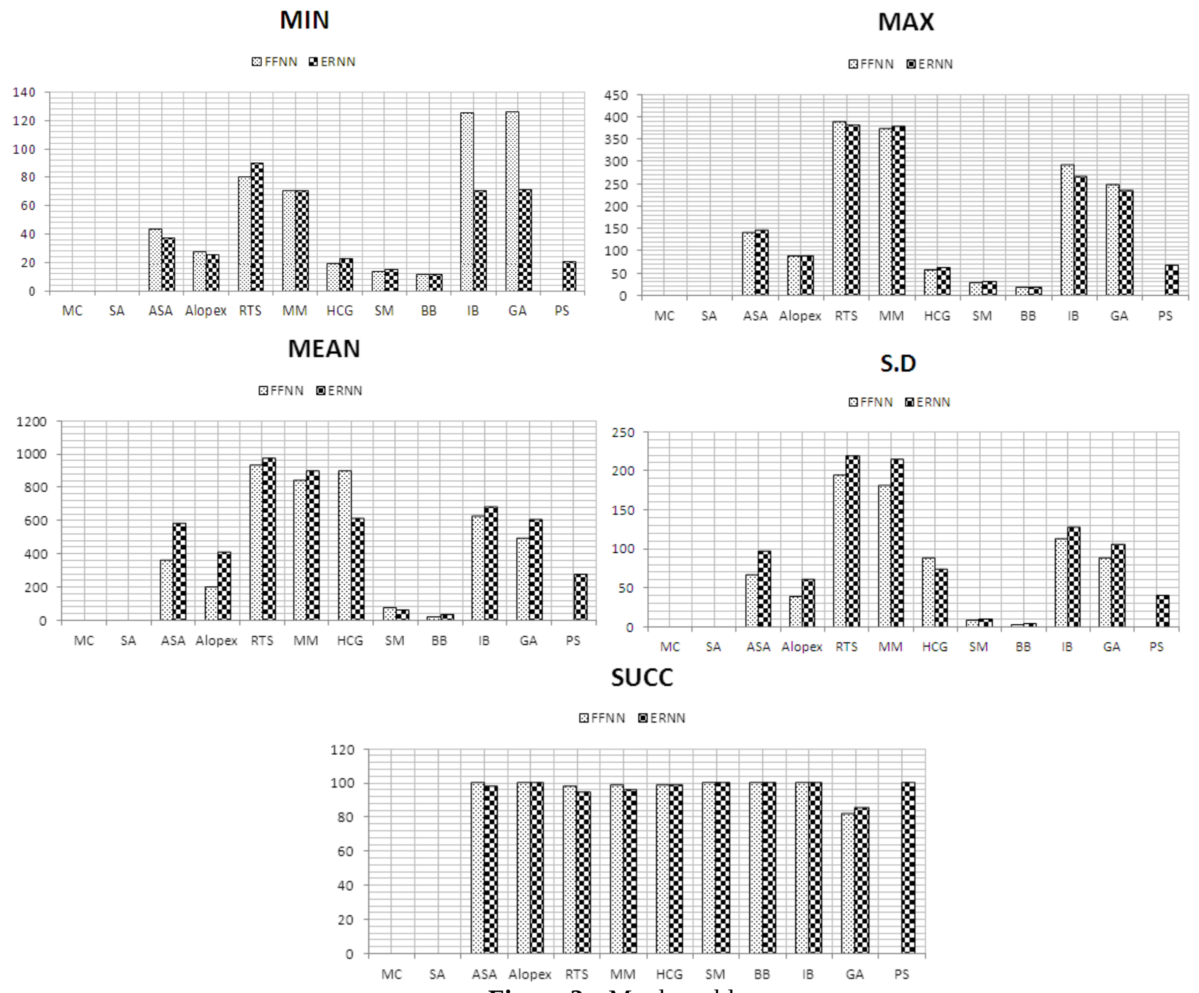

Figure 3:- Monk problem.

Generalization performance is a high impact factor in selecting an optimization algorithm. There are various techniques to prevent over-fitting and improve performance of an algorithm. As reported by previously conducted studies, these techniques have been fruitful in most of applications. However, their success highly depends on user's selection of proper algorithm and proper regulation. Also, heuristic parameters optimization depends on primary weights or optimization set size.

Performance of all algorithms strongly depends on the architecture of network. As expected, all algorithms show better generalization results by optimizing a recurrent neural network. As revealed, PSO is an excellent explorative algorithm which can properly classify most of input schemes for each variable model in Monk problems.

Regarding the generalizability of quasi-Newton algorithms, it should be noted that B&B has a better performance than other quasi-Newton algorithms. Furthermore, obtained results revealed that integrating monotone strategies in each optimization algorithms can significantly improve their performances.

Discussion and Conclusion:-

In the present study, we embedded several optimization algorithms but most of methods have been described in mathematical research articles. All of these methods cannot be appropriate for neural networks optimization. Cutting planes, successive approximation, and successive partitioning methods are generally used to make optimization concave [145]. The way of applying these methods in neural networks is not clear; so, they have been omitted here.

In this technical report, we investigated the performance of two proposed algorithms, compared them with various descending gradient methods and analyzed their modification and change to optimize two different types of neural networks in classic artificial intelligence problems.

In general, self-scaled HCGand conjugate gradient algorithms showed the best optimization performance. Regarding self-scaled non-monotone HCG which was adaptive and modified, numerical experiments showed that applying scaling factor [28] instead of scaling factor [26] has more effect on the performance of classic HCG algorithm in combination with non-monotone adaptive strategy.

Regarding conjugate gradient algorithms, we found that adaptive spectrum conjugate gradient algorithm has a higher probability in successful optimization and needs less computational cost. Moreover, it leads to quite desirable solution such that the final weight vectors presented averagely was improved in terms of generalizability and had no need to heuristic parameters with fine tuning which depend on problem, e.g. learning rate. Therefore, we concluded that scaling gradient by scaling parameter [5] and integrating with an adaptive non-monotone strategy is desirable and satisfactory.

Further, we revealed that the performance of the second-order algorithms, especially conjugate gradient algorithms, have been balanced using line search. Using various and more advanced techniques of line search [11, 13 and 34] leads to a significant improvement in the performance of these algorithms [34].

The simulation results showed that non-monotone strategy and adaption with various learning rates significantly influence practical behavior of back propagation and conjugate gradient algorithms. Accordingly, we observed a significant improvement in the performance of classic computational algorithms, more stable learning, and higher probability for a acceptable performance.

Quantitative methods in total optimization in neural networks were and resulted in various outcomes. There are challenging problems in testing total optimization algorithms. In some of these tests, local optimum numbers have a high exponential growth.

Ingber and Rosen [38] compared SA and genetic algorithms in functions with local optimums of 10^{50}. However, since non-linear optimization problems are significantly different with each other, a method which works well for some problems may do not show a good performance for other problems. Since there are few studies on the application of these methods in neural systems, it is difficult to select one of these methods as a superior one.

Unfortunately, there is no available systematic comparison of these methods. Ingber [24 and 38] compared ASA and GA I a set of problems which are usually used to test genetic methods. In all cases, ASA had a better performance over GA.

Another comparison with GENECOP in Michaleicz was performed on a set of Colville problems in which SAS showed a better performance. Comparing ASA with other simulated annealing methods also showed the superiority of this method. However, since both ASA and GA have many parameters, it is very soon to conclude that ASA is a better method.

Tabu search has been employed in various problems [43]. Dynamic hill climbing (DHC) is another interesting optimization method which seems has convergence results consistent with simulated annealing method [24].

Compared to other multistart gradient methods, GA rarely has better performance [147 and 148]. RasID algorithm has been used in only one real problem so far [48] which had a better back propagation performance in adaptive optimization and momentum amount.

The results of NOVEL [20] and applying some other optimization methods in two spiral problem and several problems were compared. These comparisons included simulated annealing (SIMANN from Netlib library but without adaptive SA), two genetic algorithms of GENOCOP (for Michalewicz [11]) and LICE (for LICE [15]), GRAND-MS using several random initialization points following descending gradient, TN-MS (the shortened Newton method with multistarts, and neural algorithm of cascade correlation constructive with random initialization).

Better results were obtained within less than 20 hours of computations on Sun SS 20/71 station [20] that was presented for 3 to 6 hidden units or 18 to 42 adaptive parameters.

In all the cases, NOVEL algorithm with accurate results (80% to 100%) in test set found the best results and then, SIMANN method was placed at the least interval. We can conclude that ASA employment in an identical problem can be followed with better performance since ASA had a significant progress in standard simulated annealing.

TN-MS also achieved the best third results. The test results were below 90% for 6 hidden neurons. Cascade correlation was placed at the fourth order such that it has 75% accurate response for 6 hidden neurons; however, only for 3 hidden neurons, only 20% was obtained while NOVEL and SIMANN obtained 80%.

Genetic algorithms obtained the worst results such that in all cases, the results were below 60%; therefore, they could not find a good solution. NOVEL on Sonar, Vovel, NetTalk optimization set, and parity-10 of UCI were also tested using hidden units. As a result, they achieved very good results and placed before TN-MA only in one case.

According to these comparisons, we conclude that genetic algorithms are often combined with neural systems and usually, they cannot be the best solution for total optimization problem and neural architectures optimization. Still, there are few experiences of other methods and most of total optimization models have not been yet tested. Undoubtedly, in near future, total optimization methods employment in neural systems will find a great importance and our studies can pave the way for dynamic researches in this regard.

References:-

1. C. Bishop, *Neural networks for pattern recognition*. Clarendon Press, Oxford, 1995.
2. S. Haykin, *Neural networks: a comprehensive foundations*. MacMillian, 1994.
3. J. Zurada, *Introduction to artificial neural systems.* West Publishing Company, St Paul, 1992.
4. B.D. Ripley, *Pattern recognition and neural networks.* Cambridge University Press, 1996.
5. R. Rojas, *Neural networks. A systematic introduction.* Springer, 1996.
6. M.H. Hassoun, *Fundamentals of artificial neural networks.* MIT Press, 1995.
7. W. Duch, R. Adamczak and N. Jankowski, Initialization and optimization of multilayer perceptrons, In *Third Conference on Neural Networks and Their Applications*, pp. 99-104, Kule, Poland, Oct. 1997; Initialization of adaptive parameters in density networks, ibid, pp. 105-110.
8. W. Duch and R. Adamczak, Statistical methods for construction of neural networks. In *International Congress on Neural Information Processing*, pp. xxx-yyy, Kitakyushu, Japan, Oct. 1998.
9. W. Duch, K. Grudzi´nski and G.H.F. Diercksen, Minimal distance neural methods. In *World Congress of Computational Intelligence*, pp. 1299-1304, Anchorage, Alaska, IJCNN'98 Proceedings, May 1998.
10. J. Schmidhuber and S. Hochreiter, Guessing can outperform many long time lag algorithms. *Technical Note* IDSIA-19-96, 1996.
11. Z. Michalewicz, *Genetic algorithms+data structures=evolution programs*, 3rd ed, Springer, Berlin, 1996.
12. S. Kirkpatrick, C.D. Gellat Jr. and M.P. Vecchi, Optimization by simulated annealing, *Science*, 220: 671-680, 1983.
13. C.J. Mertz and P.M. Murphy, UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-databases.
14. W. Schiffman, M. Joost and R. Werner, Comparison of optimized backpropagation algorithms, In *Proc. of the European Symposium on Artificial Neural Networks*, pp. 97-104, Brussels, 1993.
15. W. Duch, R. Adamczak and K. Gr¸abczewski, Extraction of logical rules from backpropagation networks. *Neural Processing Letters* 7: 1-9, 1998.
16. W. Duch and N. Jankowski, *New neural transfer functions*, Applied Mathematics and Computer Science, 7: 639-658, 1997.
17. R. Horst and P.M. Pardalos (eds), *Handbook of global optimization*, Kluwer, Dodrecht 1995; J.D. Pinter, *Global optimization in action*, Kluwer, Dodrecht 1996.
18. Goodfellow, Ian J., Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Multi-prediction deep Boltzmann machines. In Neural Information Processing Systems, December 2013.

19. P. P. van der Smagt, Minimization methods for training feed-forward networks. *Neural Networks*, 7(1): 1-11, 1994.
20. Y. Shang and B.W. Wah, *Global optimization for neural network training*, IEEE Computer, 29: 45-54, 1996.
21. S.E. Fahlman and C. Lebiere, The Cascade-Correlation learning architecture, In *Advances in Neural Information Processing Systems*, vol. 2, Morgan Kaufmann, pp. 524-532, 1990.
22. W. Duch, R. Adamczak, K. Gr¸abczewski and G. ˙ Zal, Hybrid neural-global minimization method of logical rule extraction. *Journal of Advanced Computational Intelligence*, 1998 .
23. F-M. Dittes, *Optimization of rugged landscapes: a new general purpose Monte Carlo approach*, *Physical Review Letters*, 76: 4651-4655, 1996.
24. L. Ingberg, Simulated annealing: Practice versus theory, *J. Math. Computer Modeling*, 18: 29-57, 1993; Adaptive simulated annealing (ASA): Lessons learned, *J. Control and Cybernetics*, 25: 33-54, 1996.
25. L. Ingber, Very fast simulated re-annealing. *Mathematical Computer Modeling* 12: 967-973, 1989.
26. L. Herault, Rescaled simulated annealing. In *World Congress of Computational Intelligence*, pp. 1239-1244, IJCNN'98 Proceedings, Anchorage, Alaska, May 1998.
27. J. Engel, Teaching feed-forward neural networks by simulated annealing. *Complex Systems* 2:641-648, 1988.
28. D.H. Ackley, G.E. Hinton, T.J. Sejnowski, A learning algorithm for Boltzmann machines, *Cognitive Science*, 9: 147-169, 1985.
29. J.L. McClelland and D.E Rumelhart, *Explorations in parallel distributed processing: computational models of cognition and perception.* The MIT Press, Cambridge,MA 1986.
30. R.M. Neal, *Bayesian learning in neural networks.* Lecture Notes in Statistics vol. 118, Springer, 1996
31. Choromanska, A., Henaff, M., Mathieu, M., Ben Arous, G., and LeCun, Y. The Loss Surface of Multilayer Networks. ArXiv e-prints, November 2014.
32. L. Yuille and J. J. Kosowsky, Statistical physics algorithms that converge. *Neural Computation*, 6(3):341-356, 1994.
33. Z. He, C. Wu, J. Wang and C. Zhu, A new vector quantization algorithm based on simulated annealing. In *Proc. of 1994 Int. Symp. on Speech, Image Processing and Neural Networks*, Vol. 2:654-657, 1994.
34. K. Valkealahti and A. Visa, Simulated annealing in feature weighting for classification with learning vector quantization. In *Proc. 9th Scandinavian Conference on Image Analysis*, 2: 965-971, 1995.
35. H-S. Heon and S-L. Whan, LVQ combined with simulated annealing for optimal design of large-set reference models, *Neural Networks*, 9(2): 329-336, 1996.
36. U. Kjærulff, Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 2: 7-17, 1992.
37. C. Luonan and K. Aihara, Chaotic simulated annealing by a neural network model with transient chaos, *NeuralNetworks*, 8(6): 915-930, 1995.
38. L. Ingberg, B. Rosen, Genetic algorithms and very fast simulated reannealing: a comparison. *Journal of Mathematical and Computer Modelling*, 16: 87-100, 1992.
39. K.P. Unnikrishnan, K.P. Venugopal, Alopex: a correlation-based learning algorithm for feedforward and recurrent neural networks, *Neural Computations*, 6: 469-490, 1994.
40. S. Thrun*et al.*, *The MONK's problems. A performance comparison of different learning algorithms.* Carnegi Mellon University, Technical Report CMU-CS-91-197, 1991.
41. W. Duch, R. Adamczak and K. Gr¸abczewski, Extraction of logical rules from training data using backpropagation networks. In *The First Online Workshop on Soft Computing*, pp. 25-30, Aug. 1996; also available at http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/;W. Duch, R. Adamczak, K. Gr¸abczewski, Constrained backpropagation for feature selection and extraction of logical rules. In *First Polish Conference on Theory and Applications of Artificial Intelligence*, pp. 163-170, Łód´z, Poland 1996.
42. R. Battiti and G. Tecchiolli, The reactive tabu search, *ORSA Journal on Computing* 6: 126-140, 1994; Reactive search, a history-sensitive heuristics for MAX-SAT, *ACM Journal of Experimental Algorithmics*, 2, paper 2, 1997.
43. D. Cvijovi´c and J. Klinowski, Taboo search: an approach to the multiple minima problem, *Science*, 267: 664-666, 1995.
44. R. Battiti and G. Tecchiolli, Training neural nets with the reactive tabu search, *Transactions on Neural Networks*, 6: 1185-1200, 1995.
45. H.V. Gupta, K. Hsu and S. Sorooshian, Superior training of artificial neural networks using weight-space partitioning, In *Proc. of International Conference on Neural Networks*, pp. 1919-1923, Houston, USA, 1997.
46. A.H.G. RinnooyKan and G.T. Timmer, A stochastic approach to global optimization, *American Journal of Mathematics and Management Sciences*, 4: 7-40, 1984.

47. N. Baba, A new approach for finding the globalminimum of error function of neural networks,*Neural Networks*, 2: 367-373, 1989.
48. N. Baba, Y. Moogami, M.A. Kohzaki, Y. Shiraishi and Y. Yoshida, A hybrid algorithm for finding the global minimum of error function of neural networks and its applications, *Neural Networks* 7: 1253-1265, 1994.
49. Qi-L. Liang, Z. Zhou and Z-M. Liu, A new approach to globalminimum and its applications in blind equalization, In *Proc. of International Conference on Neural Networks*, pp. 2113-2117, 199
50. R.A. Rosario *et al.*, A rapid multi-layer perceptron training algorithm, *Proc. of International Conference on Neural Networks*, pp. 824-829, Baltimore, Maryland, USA, 1992.
51. Goodfellow, Ian J., Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. In Dasgupta, Sanjoy and McAllester, David (eds.), International Conference on Machine Learning, pp. 1319–1327, 2013
52. J. Chao, W. Ratanasuwan and S. Tsuji, A new global optimization method: “Rolling-Stone Scheme” and its applications to supervised learning of multi-layered perceptrons, In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks*, Vol.2, pp. 395-398, North-Holland, Amsterdam 1992.
53. J.T.-H. Lo, *A new approach to global optimization and its application to neural networks*, In *Proc. of International Joint Conference on Neural Networks*, pp. 600-602, Baltimore 1992.
54. More information on the *PSCG* method and the implementation of the method in the SNNS simulator may be found at: http://cui.unige.ch/AI-group/staff/orsier.html
55. Z. Tang and G.J. Koehler, Deterministic global optimal fnn training algorithms, *Neural Networks*, 7: 301-311, 1994.
56. J. Hu, K. Hirasawa, J. Murata, RasID - random search method for neural network training. *Journal of Advanced Computational Intelligence*, 2(4): 134-141, 1998.
57. N. Baba, T. Shoman and Y. Sawaragi, A modified convergence theorem for random optimization method. *Information Sciences*, 13: 159-166, 1977.
58. L. Piela, J. Kostrowicki and H.A. Szeraga, The multiple-minima problem in conformational analysis of molecules. Deformation of the potential energy hypersurface by the diffusion equation method. *Journal of Physical Chemistry*, 93: 3339-3346, 1989.
59. C. Simmerling and R. Elber, Hydrophobic “collapse” in a cyclic hexapeptide: computer simulations of CHDLFC and CAAAC in water, *Journal of American Chemical Society*, 116, 2534-2547, 1994.
60. Roitberg and R. Elber, Modeling side chains in peptides and proteins: application of the locally enhanced sampling and the simulated annealing methods to find minimum energy conformations, *Journal of Chemical Physics*, 95: 9277-9287, 1991.
61. K.A. Olszewski, L. Piela and H.A. Scheraga, Mean Field Theory as a tool for intramolecular conformal optimization. Tests on terminally-blocked alanine and met-enkephalin, *Journal of Physical Chemistry*, 96: 4672-4676, 1992.
62. P.H. Winston, *Artificial intelligence, 3rd ed*, AddisonWesley, 1995.
63. R.J. van Iwaarden, *An improved unconstrained global optimization algorithm*, PhD thesis in applied mathematics, University of Colorado, 1996.
64. E. Hansen, *Global optimization using interval analysis*, Dekker, New York 1992.
65. H. Ishibuchi, H. Tanaka and H. Okada, An architecture of neural networks with interval weights and its application to fuzzy regression analysis. *Fuzzy Sets and Systems* 57:27-40, 1993.
66. Dauphin, Yann N, Pascanu, Razvan, Gulcehre, Caglar, Cho, Kyunghyun, Ganguli, Surya, and Bengio, Yoshua. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Ghahramani, Z.,Welling, M., Cortes, C., Lawrence, N.D., andWeinberger, K.Q. (eds.), Advances in Neural Information Processing Systems 27, pp. 2933–2941. Curran Associates, Inc., 2014.
67. D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, 1989.
68. R.F. Albrecht, C. R. Reeves and N. C. Steele (eds), *Artificial neural nets and genetic algorithms*, Springer Verlag, 1993.
69. ICANNGA – *Proc. of the Intern. Conference on Artificial Neural Networks and Genetic Algorithms*, 1993, 1995, 1997.
70. Y. Le Cun, *Modélesconnexionistes de l’apprentissage.* PhD thesis, Paris, 1987.
71. D.E. Rumelhart, G.E. Hinton and R.J. Wiliams, Learning internal representations by error propagation. *ParallelDistributed Processing.* MIT Press, vol. 1: 318-362, 1986.
72. D.M. Skapura, *Building neural networks*, Addison-Wesley, 1996.
73. D.B. Fogel, L.J. Fogel and V.M. Porto, Evolving neural networks, *Biological Cybernetics* 63: 487-493, 1990.

74. J. Korczak and E. Dizdarevic, Genetic search for optimal neural network, In *Conf. On Neural Networks and Their Applications*, pp. 30-45, Kule, Poland, Oct. 1997.
75. W. Schiffmann, M. Joost and R. Werner, Application of genetic algorithms to the construction of topologies for multilayer perceptrons, In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pp. 675-682, 1993.
76. X. Yao, A Review of evolutionary neural networks, *International Journal of Intelligent Systems*, 8(4): 539-567, 1993.
77. J.W. Boers and H. Kuiper, *Biological metaphors and the design of modular artificial neural networks.* Masters Thesis, Departments of Computer Sciences and Experimental and Theoretical Psychology. Leiden University, Netherlands, 1992.
78. D.J. Chalmers, The Evolution of learning: an experiment in genetic connectionism, In *Proceedings of the 1990 Connectionist Models Summer School*,Morgan Kaufmann, pp. 81-90, 1990.
79. S.A. Harp, T. Samad and A. Guha, Toward the genetic synthesis of neural networks, *Third International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 360-369, 1989.
80. M. Mandischer, Representation and evolution of neural networks, In R.F. Albrecht, C.R. Reeves and U.C. Steele (eds), *Artificial Neural Nets and Genetic Algorithms*, pp. 643-649, 1993.
81. V. Maniezzo, Genetic evolution of the topology and weight distribution of neural networks, *IEEE Transactions on Neural Networks*, 1(5): 39-53, 1994.
82. Saxe, Andrew M., McClelland, James L., and Ganguli, Surya. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In ICLR, 2013.
83. D.J. Montana, Neural network weight selection using genetic algorithms, In S. Goonatilake, S. Khebbal, editors, *Intelligent Hybrid Systems*, pp.85-104,Wiley, New York, 1995.
84. R. Salustowicz, *A Genetic algorithm for the topological optimization of neural networks*, Diplomarbeit TU Berlin, 1995.
85. D.J. Schaffer, D. Whitley and L. Eshelman, Combinations of genetic algorithms and neural networks: a survey of the state of the art, In *Proc. of the Conf. on Combination of Genetic Algorithms and Neural Networks*, pp. 1-37, 1992.
86. B. Sendhoff and M. Kreutz, Evolutionary optimization of the structure of neural network using a recurrent mapping as encoding, In *Proc. of the 3rd International Conference on Artificial Neural Networks and Genetic Algorithms*, Springer Verlag, 1997.
87. D. W. Pearson, N. C. Steele and R. F. Albrecht, Artificial neural nets and genetic algorithms. In *Proceedings of the International Conference*, Springer-Verlag, 1995.
88. D.J. Schaffer, R.A. Caruana and J. Eshelmani, Using genetic search to exploit the emergent behavior of neural networks, In S. Forest, editor, *Emergent Computation*, North Holland, pp. 244-248, 1990.
89. F.Z. Brill, D.E. Brown and W.N. Martin, Fast genetic selection of features for neural network classifiers, *Transactions on Neural Networks*, 3(2): 324-328, 1992.
90. G.W. Game and C.D. James, The application of genetic algorithms to the optimal selection of parameter values in neural networks for attitude control systems, In *IEE Colloquium on 'High Accuracy Platform Control in Space'*, pp. 3/1-3/3, Digest No. 1993/148, IEE, London, 1993.
91. D.L. Prados, New learning algorithm for training multilayer neural networks that uses genetic-algorithm techniques, *Electronics Letters*, 28(16): 1560-1561, 1992.
92. M. Srinivas and L.M. Patnaik, Learning neural network weights using genetic algorithms - improving performance by search-space reduction, In *International Joint Conference on Neural Networks*, Vol. 2, pp. 187-192, 1991.
93. D.Whitley, T. Starkweather and C. Bogart, Genetic algorithms and neural networks: optimizing connections and connectivity, *Parallel Computations*, 14(3): 347-361, 1990.
94. B. Zhang and G. Veenker, Neural networks that teach themselves through genetic discovery of novel examples, In *Proceedings of the International Joint Conference on Neural Networks*, pp. 690-695, 1991.
95. P. Arena, R. Caponetto, I. Fortuna and M. G. Xibilia, MLP optimal topology via genetic algorithms, In R.F. Albrecht, C.R. Reeves and N.C. Steele, editors, *Proceedings of the International Conference on Artificial NeuralNets and Genetic Algorithms*, Springer-Verlag, pp. 670-674, 1993.
96. R.K. Belew, J. McInerney, and N.N. Schraudolph, Evolving network: using the genetic algorithm with connectionist learning, In C.G. Langton, C. Taylor, J.D. Farmer and S. Rasmussen, editors, *Artificial Life II*, Redwood City, CA:Addison-Wesley, 1991.
97. H. Kwasnicka and P. Szerszon, NETGEN - evolutionary algorithms in designing artificial neural networks, In *Conf. On Neural Networks and Their Applications*, Kule, Poland, pp. 671- 676, 1997.

98. P. Robbins, A. Soper and K. Rennolls, Use of genetic algorithms for optimal topology determination in back propagation neural networks, In *Proceedings of the International Conference on Artificial Neural Networks andGenetic Algorithms*, pp. 726-730, 1993.
99. Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.
100. S.G. Romaniuk and L.O. Hall, SC-net: a hybrid connectionist symbolic system, *Information Sciences*, 71: 223-268, 1993.
101. J. Azevedo, *AGWIN - Manual d'utilisateur*, http://lsiit.u-strasbg.fr, Technical Report, LSIIT, Louis Pasteur University, Strasbourg, 1996.
102. E. Blindauer, *Méthodesd'encodagegénétique de réseauxneuronaux,* MSc thesis in Computer Science, Louis Pasteur University, Strasbourg, 1998.
103. E. Dizdarevic, *Optimisationgénétique de réseaux,* MSc thesis in computer science, Louis Pasteur University, Strasbourg, 1995.
104. J. Korczak and E. Dizdarevic, *Genetic optimization of neural networks*, Technical Report, CRI, Louis Pasteur University, Strasbourg, 1994.
105. The bibliography on evolutionary design of neural architectures:http://liinwww.ira.uka.de/bibliography/Neural/edna.html
106. L. Marti, Genetically generated neural networks, I. Representational effects, In *Proc. of the Intern. Joint Conference on Neural Networks*, Vol. 4, pp. 537-542, 1992.
107. D. Elizondo, *The recurrent deterministic perceptron and topology reduction strategies for neural networks*, PhD thesis, Louis Pasteur University, Strasbourg, 1997.
108. E. Fiesler, Comparative bibliography of ontogenic neural networks, In *Proceedings of the International Conference on Artificial Neural Networks*, Springer Verlag, pp. 793-796, 1994.
109. T. Ash, *Dynamic node creation in backpropagation networks,* Technical Report, Institute of Cognitive Science, University of California, 1989.
110. J.P. Nadal, Study of a growth algorithm for a feedforward neural network, *International Journal of Neural Systems*, pp. 55-59, 1989.
111. M. Hagiwara, Novel back propagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection, In *Proceedings of the International Joint Conference on Neural Network*, Edward Brothers, pp. 625-630, 1990.
112. Y. Hirose, K. Yamashita, S. Hijaya, Back-Propagation algorithm which varied number of hidden units, *NeuralNetworks*, 4(1): 61-66, 1991.
113. V. Honavar and L. Uhr, A Network of neuron-like units that learns to perceive by generation as well as reweighting of its links, In *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, pp. 472-484, 1988.
114. M. Frean, The Upstart algorithm: a method for constructing and training feedforward neural networks, *NeuralComputation*, 2: 198-209, 1990.
115. B. Bonnlander and M.C. Mozer, Latticed RBF networks: an alternative to constructive methods, In *Advances in Neural Information Processing Systems*, vol. 5, Nature and Synthetic, 1993.
116. Y. Chauvin, A Back-Propagation algorithmwith optimal use of hidden units, In *Advances in Neural Information Processing Systems*, vol. 1, Morgan Kaufmann, pp. 519-526, 1989.
117. N. Kadaba and N.E. Nygard, Improving the performance of genetic algorithms in automated discovery of parameters, In *Proceedings of the Seventh International Conference on Machine Learning*, Morgan Kaufmann, pp.140-148, 1990.
118. N. Kadaba, N.E. Nygard and P.L. Juell, Integration of adaptive machine learning and knowledge-based systems for routing and scheduling applications, *Expert Systems and Applications*, pp. 15-27, 1991.
119. D. Murray, Tuning neural networks with genetic algorithms, *AI Expert*, June 1994.
120. J.D. Kelly and L. Davis, Hybridizing the genetic algorithms and the k-nearest neighbors classification algorithms, In *Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 377-383, 1991.
121. E.J. Chang and R.P. Lippman, Using genetic algorithms to improve pattern classification performance, In *Advances in Neural Information Processing*, Vol. 3, Morgan Kaufmann, pp.797-803, 1991.
122. T. Drabe, W. Bressgott and E. Bartscht, Genetic task clustering for modular neural networks. In *Proc. of Int.Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing*, NICROSP 1996, pp. 339-347.

123. Z. Guo, R.O. Uhrig, Use of genetic algorithms to select inputs for neural networks, In *Proc. on Combination of Genetic Algorithms and Neural Networks*, pp.223-234, 1992.
124. S.A. Billings and G. L. Zheng, Radial Basis Function network configuration using genetic algorithms, *Neural Networks* 8(6): 877-890, 1995.
125. B. Burdsall and C. Giraud-Carrier, GA-RBF: A Self-optimizing RBF network, In *Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'97)*, Springer-Verlag, pp. 348–351, 1997. URL: http://www.cs.bris.ac.uk/Tools/Reports/Bibtexs/1997-burdsall-0.bib
126. M. Zamparelli, Genetically trained cellular neural networks, *Neural Networks* 10(6): 1143-1151, 1997.
127. D. W. Opitz and J. W. Shavlik, Genetically refining topologies of knowledge-based neural networks, In *International Symposium on Integrating Knowledge and Neural Heuristics*, pp. 57-66, 1994.
128. V.W. Porto, D.B. Fogel and L.J. Fogel, Alternative neural network training methods, *IEEE Expert*, pp. 16-21, June 1995.
129. P.J. Angeline, G.M. Saunders and J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, *Transactions on Neural Networks*, 5: 54-65, 1994.
130. X. Yao and Y. Lin A new evolutionary system for evolving artificial neural networks, *Transactions on Neural Networks*, 8(3): 694-713, 1997.
131. S. Smolander and J. Lampinen, Determining the optimal structure for multilayer self-organizing map with genetic algorithm. In J. Parkkinen and A. Visa, editors, *Proc. of the 10th Scandinavian Conference on Image Analysis*, pp. 411-417, 1997.
132. H. Mühlenbein, Limitations of multilayer perceptron networks - steps towards genetic neural networks, *Parallel Computing*, 14: 249-260, 1990.
133. H. Kitano, Designing neural networks using genetic algorithms with graph generation system, *Complex Systems*, 4: 461-476, 1990.
134. F. Gruau Genetic synthesis of modular neural networks, In S. Forrest, editor, *Genetic Algorithms: Proceedings of the 5th International Conference*, Morgan Kaufman, 1993.
135. J.E. Baker, Reducing bias and inefficiency in the selection algorithm. In J.Grefenstette, editor, *Proc. of theSecond Intern. Conference on Genetic Algorithms*, Los Altos, Morgan Kaufmann, pp. 14-21, 1987.
136. D. Beasley, D.R. Bull, R. Martin, An overview of genetic algorithms: Part 1, Fundamentals; Part 2, Research topics. *University Computing*, 1:(2-4), 58-69; 170-181, Cardiff, 1993.
137. T. Blickle, L. Thiele, *A comparison of selection schemes used in evolutionary algorithms*, Technical Report, ETH Zurich, 1997.
138. F. Gruau, *Neural networks synthesis using cellular encoding and the genetic algorithm*, PhD thesis, LIP, EcoleNormaleSuperieure, Lyon, 1992.
139. J. Korczak*et al.*, ECO. Research Report, LSIIT, Louis Pasteur University, Strasbourg, 1997.
140. R. Reed, Pruning algorithms - a survey, *IEEE Transactions on Neural Networks* 4(5): 740-746, 1993.
141. M. Ishikawa, Structural learning with forgetting, *Neural Networks* 9: 509-521, 1996.
142. D.J. MacKay, A practical Bayesian framework for backpropagation networks, *Neural Computations* 4: 448-472, 1992.
143. P. Cheesman and J. Stutz, Bayesian classification (AutoClass): theory and results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, editors, *Advances in Knowledge discovery and Data Mining*, pp. 153-180, MIT Press 1996.
144. D.E. Goldberg, *A note on Boltzmann tournament selection for genetic algorithms and population-orientedsimulated annealing*, *Complex Systems* 4: 445-460, 1990.
145. R. Horst and H. Tuy, *Global optimization*, Springer Verlag, 1990.
146. W. Duch, Scaling properties of neural classifiers. In *Third Conference on Neural Networks and Their Applications*, pp. 189-194, Kule, Poland, October 1997.
147. M. Mitchell, J.H. Holland and S. Forrest, When will a genetic algorithm outperform hill climbing, *Advances in Neural Information Processing Systems*, Morgan Kaufmann Publishers, Vol. 6: 51-58, 1994.
148. Fuels and M. Wattenberg, Stochastic hillclimbing as a baseline method for evaluating genetic algorithm. *Advances in Neural Information Processing Systems,* MIT Press, Vol 8: 430-436, 1996.
149. J. Kennedy and R.C. Eberhart, Particle swarm optimization, In *Proc. of the IEEE International Conf. on Neural Networks*, Piscataway, New Jersey, 1995, Vol. 4, pp. 1942-1948.
150. R.C. Eberhart and Y. Shi, Particle swarm optimization, In *Proc. of the International Conf. on Neural Networks and the Brain*, Beijing, China, 1998, pp. PL5-PL13.
151. J. Sprave, Linear neighborhood evolution strategies, In *Proc. of the 3-rd Annual Conf. on Evolutionary Programming,* San Diego, CA, World Scientific 1994.