Neural Process Lett (2016) 44:887–911
DOI 10.1007/s11063-016-9501-6

Improving Neuroevolution with Complementarity-Based Selection Operators

Tomás H. Maul[1]

Published online: 13 January 2016

Abstract This paper is concerned with the problem of improving the convergence properties of evolutionary neural networks, particularly in the context of hybrid neural networks that adopt a diversity of transfer functions at their nodes, i.e.: neural diversity machines. The paper explores the potential of solution complementarity, in the context of pattern recognition problems, and focuses on its incorporation in the selection process of recombination heuristics. In a pattern recognition context, complementarity is defined as the ability of different solutions to correctly classify complementary subsets of patterns. A broad set of experiments was conducted demonstrating that solution selection based on complementarity is statistically significantly better than random selection, in a wide range of conditions, e.g.: different datasets, recombination heuristics and architectural constraints. Although the experiments demonstrated the statistical significance and robustness of the effect, they also indicated that more work is required to increase the degree of the effect and to scale-up to larger and more complex datasets.

Keywords Neuroevolution · Evolutionary neural networks · Neural diversity machines · Recombination heuristics · Pattern recognition · Machine learning

1 Introduction

This paper is primarily concerned with the problem of improving the convergence properties of neuroevolutionary algorithms. The optimization of artificial neural networks (ANNs) using evolutionary algorithms has been shown to be effective [3,5,11,13,27,35,39–41], but is far from being a mature field. Many challenges remain. The difficulty of balancing the optimization of architectural properties (e.g., number of nodes, layers and modules and connectivity patterns) and the tuning of connection weights, although partially addressed (see for

✉ Tomás H. Maul
Tomas.Maul@nottingham.edu.my

[1] School of Computer Science, University of Nottingham Malaysia Campus, 43500 Semenyih, Malaysia

example [19]), is still largely unresolved. This issue is sometimes encapsulated by the term topology and weight evolving ANNs [35]. The fact that biological neural systems exhibit extensive use of neuronal diversity [23,34] (another architectural property), suggesting that we should be looking deeper into hybrid ANNs (HANNs) which make use of multiple transfer functions, such as in neural diversity machines (NDMs) [24], makes the optimisation task even more challenging. Unfortunately, these difficulties essentially translate into an overabundance of local minima and a tendency for premature convergence. This paper proposes one possible approach for placating these issues, by exploring the complementarity information inherent in machine learning problems. Briefly put, two solutions are complementary if they are effective at classifying or forming predictions for different (i.e., complementary) subsets of patterns. So far, to our knowledge, evolutionary recombination heuristics do not exploit complementarity information, and instead tend to favor selection methods based on probabilistic functions of individual fitness. This paper describes a set of experiments that investigate the impact of complementarity-based selection (for recombination heuristics) on the convergence speed of NDM.

ANN still trail far behind biological neural networks in terms of (1) architectural complexity, (2) evolutionary, developmental, learning and adaptive mechanisms, and (3) the resulting computational capabilities. Partly because of this, ANN research is still very much alive and thriving. From early beginnings in 1943 [26], the field has gone through several peaks (e.g., the discovery of the backpropagation algorithm) and troughs (e.g., the lull induced by support vector machines) and is recently experiencing a new wave of interest thanks to critical advances in the area of deep neural networks [18], reservoir computing [21] and extreme learning machines [20]. Another area that has been gaining interest recently is that of HANNs. According to Gutiérrez and Hervás-Martínez [14], hybridization is typically done along the following three dimensions: models, algorithms and data. In our incorporation of neuronal diversity in ANN [24] we are mostly concerned with the first dimension (i.e., models). Other works that have investigated this include: the work by Duch and Jankowski [7] where families of parameterized transfer functions were tested, the work by Stanley et al. [36] where networks with heterogeneous nodes were used for indirectly specifying the connectivity/weight patterns of other networks, and the work by Gutiérrez et al. [15,16] where different transfer functions (e.g., sigmoid, product and radial basis units) have been experimented within the same network.

One difficulty with ANNs that combine nodes with various exotic transfer functions and complex connectivity patterns consists of the lack of available learning algorithms. Fortunately one of the advantages of evolutionary approaches is that they can be directly applied to these types of networks, provided that suitable genotypic and phenotypic representations are adopted. Some of the earliest neuroevolutionary attempts include [2,38]. For useful reviews of this field refer to [10,40]. Some of the more recent and promising approaches include cooperative coevolution (e.g., SANE [28], ESP [12] and CoSyNE [13]), meta-learning [22] and interactively constrained neuroevolution [30]. Although many of these and related efforts have significantly broadened the field, to the best of our knowledge, there has been no work on incorporating complementarity information in the selection process of recombination heuristics.

In this paper, we adopted a standard evolutionary framework which includes traditional one-point crossover (CO) and mutation operators. The framework is enhanced by two other operators, namely: differential evolution (DE) [37] and rank-based uniform CO (RBUC) [33]. Note that throughout the paper we often refer to our implementation of RBUC as *probabilistic mingling* (PM). Note also, that whenever *CO* is mentioned by itself, we are referring to traditional one-point CO. Although these heuristics are equally available in the

framework, when investigating the properties of complementarity, we focused mostly on the strongest heuristic which turned out to be RBUC. Uniform CO was first proposed by Ackley [1]. The operator has been successfully used in several different applications (e.g., [6]) and has been studied theoretically at some length (e.g., [4]). The operator involves creating a new solution, by scanning parental parameters (or alleles) one-by-one, and copying each parameter (or allele) from the best parent with probability P. Although in many studies, $P = 0.5$, meaning that both parents are equally likely to contribute a parameter (this is referred to as equiprobable uniform CO in [33]), in our study, we bias P towards the stronger solution, and therefore set $P = 0.75$.

So far, as already mentioned, we have not found any previous work that explicitly incorporates the inherent complementarity information found in machine learning problems into recombination heuristics. The complementary surrogate genetic algorithm [8] relies on a different form of complementarity. In [8], the authors do away with mutation, and modify the traditional population framework in order to provide convergence guarantees in a "recombination only" approach, much as mutation would in a traditional approach. At each generation, apart from the genetic algorithm population the algorithm guarantees the existence of a set of protected solutions (complementary surrogate set), carefully designed such that each genetic loci in the former population, has all possible allele values accessible to it, via recombination with the latter population. The complementary surrogate set, guarantees that the whole search space is always available, and plays the part that mutation would typically play, guaranteeing sufficient diversity of solutions. Although the work in [8] exploits the notion of complementarity it does not specifically seek out to pair solutions that are mutually complementary. For a useful review on recombination strategies in the context of the optimization of neural networks by genetic algorithms see [17].

So there seems to be a serious gap in the literature with regards to the application of complementarity in recombination heuristics, but how about the notion of complementarity in the first place? Has this information been effectively used in the past? Indeed, complementarity based information has been used in several different branches of machine learning, including: (1) ensemble learning methods in general, where many different measures of classifier diversity have been tested and compared, many of which explicitly incorporate notions of complementarity [22], (2) AdaBoost, where complementary sets of patterns (or pattern weights) shaped by error dynamics are used to guide the formation of ensembles of weak classifiers [32], (3) neuroevolution, where complementarity is used more from the perspective of diversity maintenance in order to avoid premature convergence [35], (4) cascade correlation neural networks, where error/pattern complementarity is used in a sequence of training stages in order to minimize the training error and build a network with a cascade topology [9]. In summary, the concept of error complementarity is widespread in machine learning, but does not seem to be used explicitly in evolutionary recombination operators.

The importance of complementary classifications (CCs) struck us when considering the problem of competing conventions [31], which is frequently tackled in the neuroevolution field. The problem of competing conventions arises due to the fact that very different phenotypes may actually be computing the same (or similar) function(s), which in turn can be problematic for recombination heuristics. Consider the following four essential features of neural networks, i.e., phenotype, error, internal computation, and input–output mapping. When comparing any two specific instantiations of networks with regards to these features, we can identify several unique and valid cases (Table 1), one of which corresponds to competing conventions (i.e., case 2), and another of which corresponds to CCs (i.e., case 4). When two networks are phenotypically different, have different internal computations and different input–output mappings but have the same error, this implies that these networks

Table 1 Phenotype: appearance of the network

Cases	Phenotype comp.	Internal	In-out mapping	Error	Comment
1	Same	Same	Same	Same	Same network
2	Different	Same	Same	Same	Competing conventions
3	Different	Different	Same	Same	Complementary computations
4	Different	Different	Different	Same	Complementary classifications
5	Different	Different	Different	Different	Completely distinct networks

Note that two networks maybe have the same error whilst having two different input–output mappings (one network can perform well in one subspace, whereas the other performs well in another subspace). *Internal computation*: how the overall function is being computed internally, *in-out mapping*: overall function being computed, *error*: overall error

must be complementary somehow. Note however, that this case does not encompass all possible complementary network pairs, some of which could also have different errors (i.e., some complementary pairs could also be included in case 5). The crucial factor for complementarity is for error patterns to be complementary (e.g., network A classifies patterns 1–3 correctly, whereas network B classifies patterns 3–6 correctly).

In conclusion, there seems to be a clear gap in the literature pertaining to the application of complementarity based information in evolutionary optimization. On the other hand, complementarity based information has been successfully used in several different contexts such as ensemble learning, which proves its viability and potential. Our goal in this paper is to show how error complementarity in the context of pattern recognition can be advantageously incorporated into recombination heuristics for the evolutionary optimization of neural networks. Since the notion of complementarity can be broadened to other domains beyond pattern recognition, we believe this research may help improve the convergence properties of other optimization algorithms and problems.

The next section provides an overview of the methods adopted, including the neural network approach adopted, the evolutionary algorithms used, the concept of CCs and the experimental design. Subsequent to that, the results are described, followed by a discussion of their implications, the strengths and weaknesses of the approach and future work.

2 Methods

2.1 Neural Diversity Machines

The neuroevolutionary experiments reported in this paper were all conducted on a specific type of HANN, denoted NDMs, which was first reported in [24] and is characterized by relatively large pools of weight (or activation) and node (or output) functions. Refer to Table 2 for a complete list of the functions available to network nodes. In this framework architectures are allowed to be recurrent, and as observed in [24], architectures evolved for particular problems do indeed tend to be recurrent, i.e., strictly feedforward architectures are rarely evolved. Note that several experiments control the size of the pool of available functions, i.e.: weight function, WF (or node function, NF) set cardinality. If for instance, for a particular optimization experiment, we restrict WF cardinality to 3, this means that at

Table 2 NDM weight and node functions

Numbers	Types	Functions	Equations
1	WF	Inner product	$a_j^{(t+1)} = b_j + w_{j0}a_j^{(t)} + \sum_{i=1}^{n} w_{ji}x_i^{(t)}$
2	WF	Euclidean distance	$a_j^{(t+1)} = \sqrt{\sum_{i=1}^{n}(w_{ji} - x_i^{(t)})^2}$
3	WF	Higher-order product	$a_j^{(t+1)} = \prod_{i=1}^{n} cw_{ji}x_i^{(t)}$
4	WF	Higher-order difference	$a_j^{(t+1)} = \sum_{i=2}^{n} \lvert x_1^{(t)} - x_i^{(t)} \rvert$
5	WF	Weighted deviation	$a_j^{(t+1)} = \left(\sum_{i=1}^{n} w_i\right)\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i^{(t)} - \bar{x}^{(t)})^2}$
6	WF	Minimum	$a_j^{(t+1)} = min[w_{j1}x_1^{(t)}, \ldots, w_{jn}x_n^{(t)}]$
7	WF	Maximum	$a_j^{(t+1)} = max[w_{j1}x_1^{(t)}, \ldots, w_{jn}x_n^{(t)}]$
1	NF	Identity	$o_j = a_j$
2	NF	Sigmoid	$o_j = \frac{1}{1+e^{-ca_j}}$
3	NF	Threshold Gaussian	$o_j = e^{-\frac{(a_j)^2}{c}}$ if $o_j \geq d$ then $o_j = 1$

WF and NF denote weight and node functions, respectively, t denotes time, i and j index network nodes, a denotes the result of an activation (or weight) function, o denotes the result of applying an output (or node) function, b denotes a bias, w refers to weights, c denotes a constant, n denotes the number of inputting nodes and x denotes the activation of an inputting node

the start of each optimization process, three WFs are randomly selected and fixed as the only WFs available to networks.

For most experiments a compact 2–4–1 architecture was adopted (i.e., two input nodes, four middle or hidden nodes, and one output node). Although this architecture is quite small, the fact that nodes could adopt any one of 21 functions (because 7 WFs * 3 NFs = 21 transfer functions), significantly compensated for this compactness. Furthermore, the fact that networks improved their performance through evolution and exhibited consistent behavioural differences according to specific experimental conditions, were both suggestive of the adequacy of these network sizes. For a full list of the NDM parameters adopted in the paper's experiments refer to Table 3.

2.2 Evolutionary Algorithm

Algorithm 1 provides a high-level summary of the evolutionary algorithm adopted in this paper. From lines 2 to 5, the evolutionary and NDM parameters are defined and the solutions (i.e., different network configurations) are initialized, evaluated and sorted. At line 7 the basic generational loop is setup where stopping conditions are determined by standard parameters such as maximum number of generations (or evaluations) and target cost. The first step in a generation involves expanding the current population by applying several heuristics [i.e., one-point CO, PM (or RBUC), mutation and DE]. The new solutions in this expanded population are then evaluated, sorted, trimmed, and finally re-expanded with random solutions if necessary. Evaluation and sorting are performed on this population and finally stopping conditions are checked.

Algorithm 2 summarizes the function used for evaluating a single solution. For each raw solution (i.e., a vector of values each within the range [0, 1]) that is passed to it, the function

Table 3 NDM parameters adopted in all experiments reported in this paper, unless otherwise stated

IDs	Description	Parameters
P1	Number of input nodes	2
P2	Number of middle nodes	4
P3	Number of output nodes	1
P4	Maximum iterations	5
P5	Minimum weight	−1
P6	Maximum weight	1
P7	Minimum sigmoid steepness	0.3
P8	Maximum sigmoid steepness	3
P9	Minimum Gaussian steepness	0.01
P10	Maximum Gaussian steepness	10
P11	Minimum Gaussian threshold	0
P12	Maximum Gaussian threshold	2
P13	All nodes must exist	False
P14	All connections must exist	False
P15	Weight functions available	[1 2 3 4 5 6 7]
P16	Node functions available	[1 2 3]
P17	Inputs not recurrent	True
P18	Outputs not recurrent	True
P19	No input–output connections	False
P20	No lateral connections at middle nodes	False
P21	Force use of all weight functions	False

Most parameters are relatively self-explanatory except perhaps parameters 17–21. Parameter 17 was set to true, signifying that network nodes were not allowed to send recurrent connections back to input nodes. Parameter 18 was set to true, which means that output nodes were not allowed to send recurrent connections to other network nodes. Parameter 19 was set to false, which means that input nodes were allowed to send connections directly to output nodes. Owing to parameter 20 being set to false, middle nodes were allowed to connect to each other. If parameter 21 is set to true this means that networks must imperatively use each weight function at least once. Since in these experiments we set this parameter to false, networks were allowed to express any subset of weight functions

creates an NDM network (line 2), which it runs on each input pattern in the data set (line 6) and computes an error which measures the discrepancy between the NDM output and the target output (line 7). This error is both accumulated in the *cost* variable (line 8) and concatenated to the *errorVec* variable (line 9), the former being a standard evolutionary cost term and the latter being essential to complementarity based computations.

The CO, mutation and DE heuristics depicted in Algorithm 1 are all standard operators. CO consists of a simple one-point CO operator. Mutation is implemented by scanning each parameter of a solution and probabilistically deciding whether to mutate it, and if so, adding a random number that falls within a specified range of values. Our implementation of DE involves creating an ordered list of sufficiently different solutions, selecting pairs of solutions in order to compute differential vectors, and using these vectors in order to generate new solutions. PM, which is depicted in Algorithm 3, creates a new solution by selecting two different indices (line 6) from the total number of solutions in the population that was passed as an argument (line 1). These indices can either be random or based on complementarity

Algorithm 1 High-level summary of the evolutionary algorithm adopted.

```
1: procedure EVOLVE(data)
2:     pEvol ← set evolutionary parameters;
3:     pNDM ← set NDM parameters;
4:     solutionsI ← initSolutions(pEvol, data);                    ▷ Initialize solutions
5:     solutionsS ← evaluateSort(solutionsI, data, pNDM);          ▷ Evaluate and sort
6:     doStop ← false;
7:     while !doStop do                                            ▷ Loop through generations
8:         coSols ← []; pmSols ← []; mSols ← []; deSols ← [];      ▷ Initialize sol. subsets
9:         if p.doCrossOver then
10:            coSols ← doCrossOver(solutionsS,pEvol);             ▷ Crossover
11:        end if
12:        if p.doProbMingle then
13:            pmSols ← doProbMingle(solutionsS,pEvol);            ▷ Probabilistic mingling
14:        end if
15:        if p.doMutation then
16:            mSols ← doMutation(solutionsS,pEvol);               ▷ Mutation
17:        end if
18:        if p.doDiffEvol then
19:            deSols ← doDifferential(solutionsS,pEvol);          ▷ Differential evolution
20:        end if
21:        solutionsG ← [solutionsS; coSols; pmSols; mSols; deSols];   ▷ Concatenate
22:        solutionsS ← evaluateSort(solutionsG, data, pNDM);      ▷ Evaluate and sort
23:        solutionsT ← trim(solutionsS, pEvol);                   ▷ Keep strong and diverse solutions
24:        solutionsX ← expand(solutionsT, pEvol);                 ▷ Add solutions if necessary
25:        solutionsS ← evaluateSort(solutionsX, data, pNDM);      ▷ Evaluate and sort
26:        doStop ← check stopping conditions
27:    end while
28: end procedure
```

Algorithm 2 Function for evaluating a single solution.

```
1: procedure EVALUATEONE(solution, data, ndmParam)
2:     ndmNet ← createNet(solution, ndmParam);                 ▷ Create the NDM defined by sol.
3:     cost ← 0;                                               ▷ Initialize the cost to zero
4:     errorVec ← [];                                          ▷ Initialize an empty vector of errors
5:     for pattern ∈ data do                                   ▷ Loop through patterns
6:         netOutput ← runNet(pattern.input, ndmNet);          ▷ Run NDM on pattern
7:         anError ← computeError(pattern.output, netOutput);
8:         cost ← cost + anError;                              ▷ Accumulate the error
9:         errorVec ← [errorVec anError];                      ▷ Concatenate errors
10:    end for
11:    cost ← cost / size(data);                               ▷ Compute average
12:    return cost, errorVec;
13: end procedure
```

information and refer to two parent solutions from which genetic material is to be recombined to form a new solution. This recombination is done based on the quality of each solution. If the cost of the first solution is lower than that of the second, then the new solution is more likely to inherit material from the first solution, and conversely if the second solution has the smallest cost (lines 11–25). The probabilistic decision of which parent to choose in order to copy a parameter is made at each parameter index (lines 18–22).

Once solutions have been expanded using different heuristics, and have been evaluated and sorted, the evolutionary algorithm in Algorithm 1, performs trimming, which fulfils the dual function of preserving the stronger solutions and maintaining diversity. Refer to Algorithm 4 for a summary of the trimming function. The function makes use of the parameter *propTrim*

Algorithm 3 Pseudo-code for the probabilistic mingling heuristic.

```
1: procedure PROBMINGLE(solutions, p)
2:     pmSols ← [];                                   ▷ Initialize PM solutions
3:     [numSol numParam] ← size(solutions);           ▷ Get num. sol. & param.
4:     for si ← 1:p.numPM do                          ▷ Create p.numPM new solutions
5:         aPMsol ← zeros(1,numParam);                ▷ Init. to zeros
6:         [soli1 soli2] ← selectSolPair(numSol,p);   ▷ Random or complementary
7:         sol1 ← solutions(soli1).raw;               ▷ Solution 1
8:         cost1 ← solutions(soli1).cost;             ▷ Cost 1
9:         sol2 ← solutions(soli2).raw;               ▷ Solution 2
10:        cost2 ← solutions(soli2).cost;             ▷ Cost 2
11:        if cost1 < cost2 then                      ▷ If cost1 is smaller than cost2 then there is ...
12:            probOne ← 0.75;                        ▷ ... a higher probab. of its param. being selected
13:        else
14:            probOne ← 0.25;                        ▷ Otherwise, lower probab. of param. being selected
15:        end if
16:        for pi ← 1:numParam do                     ▷ Do the mingling
17:            randVal ← rand;                        ▷ Generate a random number
18:            if randVal ≤ probOne then
19:                aPMSol(pi) ← sol1(pi);             ▷ Parameter copied from sol1
20:            else
21:                aPMSol(pi) ← sol2(pi);             ▷ Parameter copied from sol2
22:            end if
23:        end for
24:        pmSols ← [pmSols; aPMSol];                 ▷ Append new PM solution
25:    end for
26:    return pmSols
27: end procedure
```

which determines the maximum number of solutions in the trimmed set (line 3), e.g.: if the population contains 100 solutions and $propTrim = 0.4$, then the trimmed set will contain at most 40 solutions. The reason why the trimmed set may contain fewer than *numRequired* solutions is that the function will disregard some solutions that are too similar to other solutions already present in the trimmed set (lines 10–11) and there may be an insufficient number of sufficiently different solutions in the population. The function starts by adding the best solution in the population to the trimmed set (line 7), and then searches the population in decreasing order of quality for the next sufficiently different solution (lines 10–11). When it finds a solution it adds it to the trimmed set (line 15), and continues searching from the current position until it finds another solution that is sufficiently different from the latest solution to have been added to the trimmed set. This continues (lines 8–17) until the cardinality of the trimmed set has reached *numRequired* or there are no more solutions to search.

In line 24 of Algorithm 1, an *expand* function is invoked. This function compares the size of the trimmed set with the target size of each generation population. If the former is smaller than the latter, then the *expand* function will fill up the remaining spaces with random solutions.

2.3 Complementary Classifications

Complementarity-based evolutionary processes, as we define them here, are mostly focused on solution recombination. During recombination (e.g., CO), solution pairs are typically chosen via random selection (RS) or some probabilistic function of fitness (e.g., N-tournament and roulette wheel). The complementarity concept tries to improve the solution selection step by attempting to find solutions that complement each other somehow (complementary

Algorithm 4 Pseudo-code for the population trimming function.

```
1: procedure TRIM(solutions, p)
2:     numSol ← size(solutions);                          ▷ Number of solutions
3:     numRequired ← round(numSol × p.propTrim);          ▷ Cardinality of trimmed set
4:     trimSols ← [];                                     ▷ Initialize set of trimmed solutions
5:     seli ← 1; inc ← 0; stori ← 1;                      ▷ Init. indices
6:     curSol ← solutions(seli);                          ▷ The first solution is the best
7:     trimSols ← [trimSols; curSol];                     ▷ Add current solution
8:     while (stori < numReqSol) AND (inc ≠ −1) do        ▷ While new sol. can be found
9:         remainSol ← solutions(seli+1:end);             ▷ Extract remaining solutions
10:        inc ← getNext(curSol, remainSol, p.minDiff);   ▷ Get next sufficiently ≠ sol.
11:        if inc ≠ −1 then                               ▷ If inc is −1, a sufficiently ≠ sol. was not found
12:            seli ← seli + inc;                         ▷ Increment selected index by inc
13:            stori ← stori +1;                          ▷ Increment count of stored solutions
14:            curSol ← solutions(seli);                  ▷ Extract the identified solution
15:            trimSols ← [trimSols; curSol];             ▷ Store the solution
16:        end if
17:    end while
18:    return trimSols;
19: end procedure
```

solution pair), therefore in principle yielding higher quality offspring. In this paper we often refer to this approach as CC selection (CCS). Solutions can be complementary in many ways. We are here focusing on pattern recognition problems and therefore on classification complementarity. To put it crudely, solution s_1 and solution s_2 are considered to be complementary if s_1 classifies one set of patterns correctly whereas s_2 correctly classifies a distinct set of patterns from the same dataset: together they classify a broader set of patterns correctly. On the other hand, if s_1 and s_2 classify the exact same patterns correctly then they exhibit no complementarity.

The explanation in the paragraph above hides an important distinction between complementarity in general and *useful complementarity*. Consider the example of a data-set consisting of 10 patterns, where the target output for patterns 1–5 is "0", whereas the target output for the remaining patterns is "1". Consider then a solution s_1 which always outputs "0" regardless of what the input pattern is, and a converse solution s_2 which always outputs "1". Clearly these two solutions will be fully complementary (100 % accuracy overall), but since neither of them has any discriminatory power, their recombination is extremely unlikely to produce any useful offspring. The obvious solution to this problem is of course to enforce a minimum level of discriminatory power for each solution. In order to further avoid this problem many of our complementarity-based operators also take into account how much classification overlap the different solutions exhibit. If solutions have both complementarity and some overlap, then they are likely to be useful in the context of complementary recombination.

As in [22] we use the notation N^{11} to denote the number of patterns that are correctly classified by both solutions, N^{00} to denote the number of patterns that neither one of the solutions classifies correctly, N^{10} to denote the number of patterns that are correctly classified by solution 1 and incorrectly classified by solution 2, and N^{01} to denote the number of patterns that are correctly classified by solution 2 and incorrectly classified by solution 1.

As was already mentioned, complementarity-based processes occur at the stage of selecting pairs of solutions for recombination. For the case of PM this can be seen in line 6 of Algorithm 3. For the purposes of this research we experimented with several different variations of complementarity, all of which (including those that failed) are here listed for convenience sake: see Table 4. The overall approach involves building a CCs matrix (*ccMat*),

Table 4 Summary of the different variants of complementarity experimented with

Versions	Description (summary)
ccMat1	ccError = $(N^{11} + N^{10} + N^{01})$/numPat; **if** $(N^{11} \geq 1)$ AND (err1 $\leq$ 0.499) AND (err2 $\leq$ 0.499) **then** ccMatrix(si1,si2)=ccError; **else** ccMatrix(si1,si2)=1; **end**
ccMat2	ccMatrix(si1,si2)=(sum(errPat1)+sum(errPat2))/(numPat $\times$ 2)
ccMat3 (*)	ccError=$(N^{11} + N^{10} + N^{01})$/numPat; **if** $(N^{11} \geq 1)$ AND ((err1 $\leq$ 0.499) OR (err2 $\leq$ 0.499)) **then** ccMatrix(si1,si2)=ccError; **else** ccMatrix(si1,si2)=1; **end**
ccMat4	ccMatrix(si1,si2)=min(err1,err2);
ccMat5	ccMatrix(si1,si2)=(err1+err2)/2;
ccMat6 (*)	ccQual=normFac $\times(N^{11} \times (N^{10} + N^{01}))$ ccMatrix(si1,si2)=1− ccQual;
ccMat7 (*)	ccQual=normFac $\times(N^{11} \times N^{10} \times N^{01})$; ccMatrix(si1,si2)=1− ccQual;
ccMat8	ccQual=normFac $\times Z^{11} \times (Z^{01} + Z^{10})$; ccMatrix(si1,si2)=1− ccQual;
ccMat9	ccQual=normFac $\times Z^{11} \times Z^{01} \times Z^{10}$; ccMatrix(si1,si2)=1− ccQual;
ccMat10	ccQual=normFac $\times(Z^{11} + Z^{01} + Z^{10})$; ccMatrix(si1,si2)=1− ccQual;
ccMat11 (*)	ccMatrix(si1,si2)=$(N^{11} + N^{10} + N^{01})$/numPat;

Note that all CC matrices are normalized so that all values fall within the range [0, 1], where 0 and 1 denote best and worst complementarity values, respectively. (*) Main successful variants. *si1*, *si2*: solution 1 index and solution 2 index, respectively, *errPat1*, *errPat2*: error patterns for solutions 1 and 2, respectively, *err1*, *err2*: average number of erroneous patterns by solutions 1 and 2, respectively, *normFac* is a factor that normalizes a result to fall within the range [0, 1], Z : the same as the N notation except that pattern counts are weighted by pattern difficulty

which is then used for probabilistic solution selection. The dimensionality of *ccMat* is $n \times n$ where n refers to the number of solutions in the population. The matrix is constructed by analysing the relationships between the error patterns (see line 9 in Algorithm 2) between two different solutions. Each cell indexed by (i, j) contains a value representing the complementarity between solutions i and j. Different complementarity approaches (Table 4) differ in how they define useful complementarity. Probabilistic solution selection uses a ccMat in order to select a subset of solution pairs (out of all possible pairs) which satisfy a minimum constraint on complementarity, and then selects a random pair from within this smaller set.

For each variant in Table 4 the main differentiating elements are presented. Note that for each case, the complementarity matrix building function scans over all possible pairs of solutions (not depicted). The *ccMatrix*(*si1*,*si2*) assignments depicted in Table 4 are executed

for every single possible pair of solutions and hence N^{11} refers to the number of patterns correctly classified by both solutions in the solution pair (si1, si2). The same point applies to N^{10}, N^{01}, Z^{11}, and so on.

2.4 Experimental Design

In this paper we report the results of nine experiments. The general goals of these experiments were to test the feasibility of the complementarity concept and explore some of the conditions surrounding its effectiveness, with the primary motive of designing evolutionary algorithms that derive maximal benefit from complementarity. Each one of the nine experiments was designed with a particular set of questions in mind. We summarize these experiments and questions here for the reader's convenience.

- *Experiment 1*. In the context of several simple synthetic data-sets, which recombination heuristic is most effective from the following: PM, CO and DE?
- *Experiment 2*. Can two solutions (with sufficiently different error patterns) be recombined such that the error of the best recombination is equal to or lower than the complementary error of the original pair of solutions? What effect does the "overlap of correct classifications" have on this?
- *Experiment 3*. How do different types of complementary error change as a set of randomly initialized solutions is gradually grown?
- *Experiment 4*. What is the best "CC Matrix" construction method?
- *Experiment 5*. Is complementarity based selection robust to different optimization heuristics?
- *Experiment 6*. How do random and complementarity based selection compare in terms of the temporal (generational) evolution of different measures such as the mean error of all solutions within each generation and the best complementary error of the best complementary pair in each generation.
- *Experiments 7 and 8*. Does complementarity influence the effect of NDM constraints (e.g., WF set cardinality) on convergence speed?
- *Experiment 9*. Does the complementarity based effect scale-up to larger and harder data-sets?

For the majority of experiments (from 1 to 8) 2D synthetic data-sets consisting of different black and white regions were used. These data-sets are depicted in Fig. 1. In experiment 9, several real-world data-sets from the UCI Machine Learning Repository were used [1]. The NDMs, evolutionary algorithms and experiments were all implemented from scratch using MATLAB version 7.12 by MathWorks.

3 Results

3.1 Experiment 1

In order to investigate the usefulness of complementary solutions in the context of solution recombination in neuroevolution, it was necessary to choose a simple, efficient and effective global recombination-based heuristic. Figure 2 compares three such heuristics, i.e.: PM, CO and DE. The heuristics were compared using 4 different data-sets and by running 20 tests per condition. In each test, the optimization goal was to reach a target error of 0.35 (within a maximum of 3000 evaluations) and the output of each test was the number of evaluations

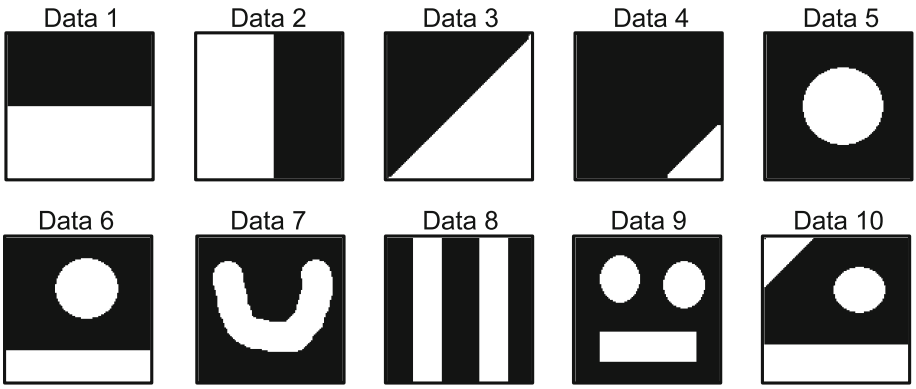

Fig. 1 Synthetic 2D data-sets used in most experiments from *1* to *8*

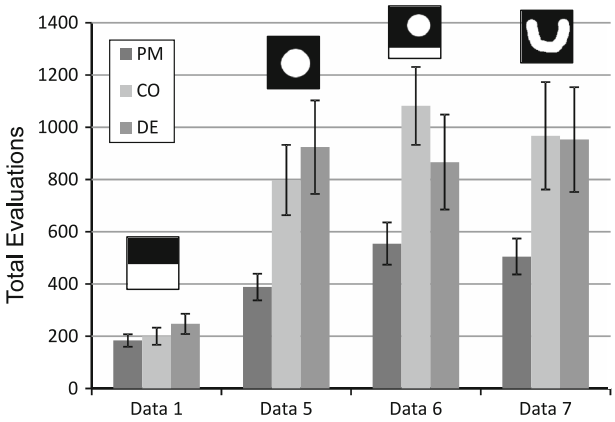

Fig. 2 Comparison of three global recombination-based heuristics

required to reach the target cost. The graph clearly shows the relative superiority of PM and therefore of its adequacy for further experimentation. According to these results, PM seems to be a good candidate for solution recombination in neuroevolution, since it seems to avoid (or at least attenuate) some of the problems manifested by traditional CO heuristics as commonly reported in the literature.

3.2 Experiment 2

Before proceeding with an explanation of the experiment it is necessary to clarify the term "deep recombination". Deep recombination is in some ways similar to PM except that it is more focused. The idea is to take two different solutions and find a third solution which optimally combines parameters from both parents. In order to do this, we optimize a vector of 0's and 1's which deterministically dictates from which parent a parameter should be copied (e.g., "0" for the first parent and "1" for the second parent).

Figure 3 attempts to answer the question: can complementary solutions be effectively recombined? More specifically, the underlying experiment compares the best (i.e., minimum) error within a pair of complementary solutions, to the error of the solution resulting from the

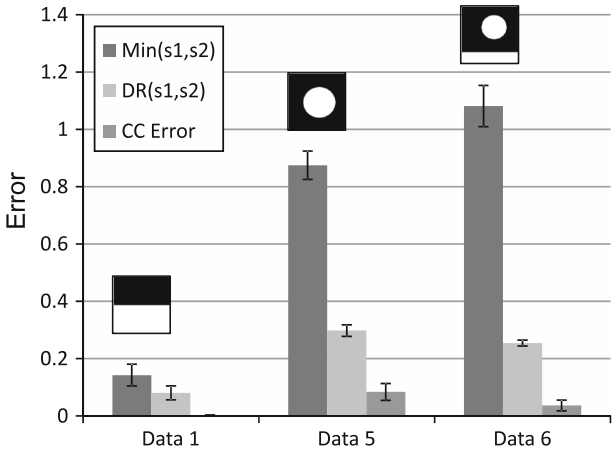

Fig. 3 Complementary solutions can be effectively recombined

deep recombination of the same pair. The term "Min(s1,s2)" refers to the smallest error of the pair of solutions whereas the term "DR(s1,s2)" refers to the error of the solution resulting from deep recombination of the solutions s_1 and s_2 and the term "CC error" denotes the complementary (i.e., overall) error of the chosen solution pairs. To further clarify the term "CC error" if a dataset consists of four patterns, and solution s_1 correctly classifies patterns 1 and 2, and solution s_2 correctly classifies patterns 2 and 3, the solution pair (s_1, s_2) correctly classifies three patterns, and therefore the "CC error" is 25 %. The actual error metric used in Fig. 3 is a bit more complicated since it incorporates a distance metric, although the general concept is the same. The figure depicts an average over 10 tests. Each test was divided in two main phases. In the first phase a solution pair with a complementary error approximating 0 (and a minimum overlap of one correct classification) was searched for, whereas in the second phase deep recombination was performed on that pair for 10 generations. The graph in Fig. 3 shows that deep recombination of solutions in a complementary pair can consistently find solutions that are better than either one of the component solutions [i.e., DR(s_1, s_2) is significantly smaller than Min(s_1, s_2)] and that this ability seems to get more pronounced as the difficulty of the data-set increases [i.e., the difference between Min(s_1, s_2) and DR(s_1, s_2) seems to increase with problem difficulty]. Note also that in spite of the feasibility of deep recombination, the resulting solutions can still be improved, i.e.: the average CC error is significantly smaller than the average deep recombination error.

3.3 Experiment 3

The experiment depicted in Fig. 4 attempted to investigate how different types of complementary errors change as we grow a set of randomly initialized solutions. This simple experiment gives an idea of the potential of complementarity in a pure context, where no optimization mechanisms are employed. The experiment also controlled the cardinality of the WF set available to networks in order to see in what way(s) the latter affected complementarity in this context. For each condition, 100 tests were run and then averaged to obtain the curves seen in the figure. The x-axes of both sub-figures depict the number of random solutions/networks in the population (i.e., from 1 to 20). The y-axis of the left sub-figure depicts the CC error of

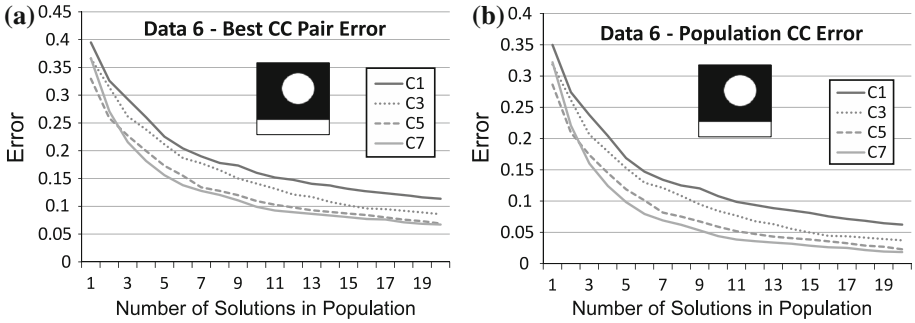

Fig. 4 The progression of complementarity errors in a growing set of randomly initialized solutions

the best solution pair in the population, whereas the y-axis of the right sub-figure depicts the CC error of the whole population of solutions. C1–7 refer to WF set cardinalities from 1 to 7.

Looking at Fig. 4 several simple points can be made. Firstly, both types of CC error (i.e., best pair and whole population) decrease with the number of random solutions. Secondly, the population CC error is lower for each number of solutions, compared to the best-pair CC error. This is to be expected since the classification complementarity of the whole population includes the one afforded by the best pair together with what is afforded by the remaining solutions. Thirdly, both types of CC error are inversely proportional to WF set cardinality. This last observation is indicative of some of the advantages of neuronal diversity. Here, neuronal diversity, as represented by WF set cardinality, decreases CC error, even when no optimization or explicit diversity maintenance mechanisms are being used. So far, Fig. 3 shows that complementarity can be exploited (e.g., through deep recombination) and Fig. 4 shows that complementarity emerges easily (i.e., even through random initialization).

3.4 Experiment 4

Having shown the potential usefulness and feasibility of complementarity (in Figs. 3, 4) the next obvious step was to use complementarity information in the recombination process in order to investigate how best to exploit it. Figure 5 illustrates several simple experiments that clearly show the merits of using complementarity information during the recombination process. Note that in this context we define the recombination process to include both solution selection and solution fusion. In the experiment embodied by Fig. 5, complementarity information was used in order to guide the selection of solution pairs for recombination. In the "Rand. Sel." condition, solution pairs for recombination were selected randomly whereas for the other conditions, the probability of selecting a particular pair of solutions was roughly proportional to their complementarity. Refer to the Sect. 2 for a more detailed explanation of these methods. For each experimental category (i.e., sub-figures a–d) depicted in Fig. 5, the objective was to compute the average number of evaluations required in order to reach a specific target error (depicted in the figure), for a particular dataset. For each category, multiple tests were run. From a quick inspection of Fig. 5, several observations can be made: (1) all CC based methods were better than RS in all conditions and (2) which particular CC based selection method is best may depend on target error and/or data-set. For instance, Fig. 5a, b suggest that, for data 6, when the target error is not too ambitious (i.e., 0.3), neither one of the CC based selection methods stands out, whereas when the target error is decreased

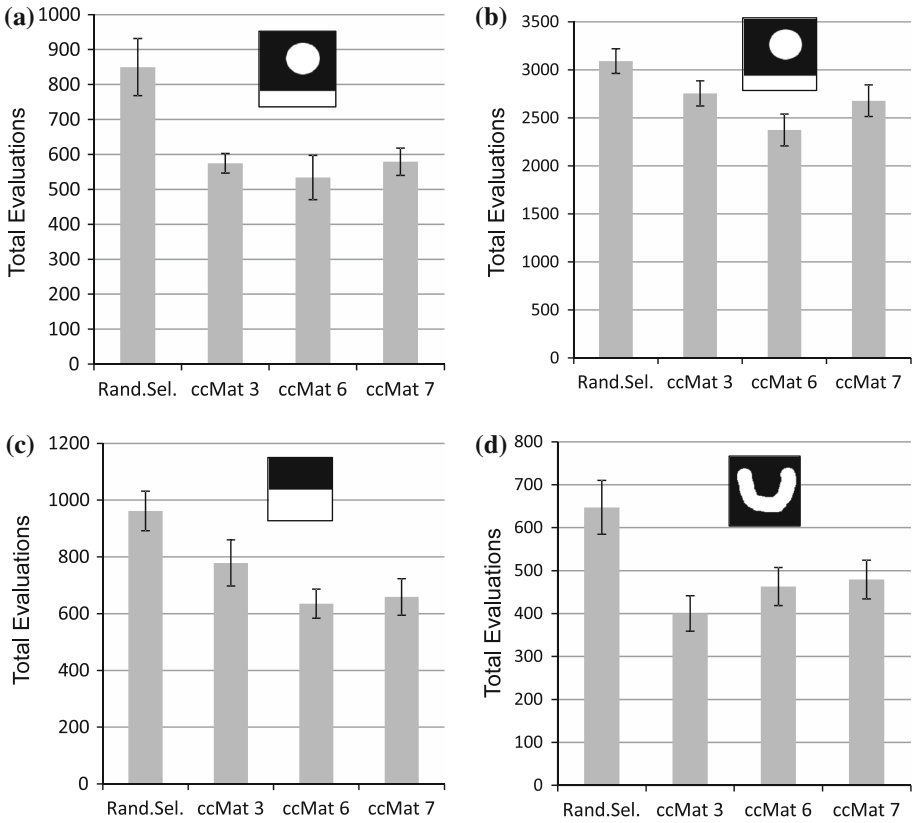

Fig. 5 The impact of complementarity based solution selection for different conditions

(i.e., 0.25), the *ccMat6* method (i.e., $N^{11}(N^{10} + N^{01})$), appears to be the best candidate. Moreover, note Fig. 5d, where the *ccMat3* method seems to be the best choice.

3.5 Experiment 5

Having established that the utilization of CC information in the solution selection stage can speed up convergence in the case of PM, the next question we addressed pertained to the generalizability of this strategy. In other words, can CC based selection improve any solution recombination heuristic? The experiment depicted in Fig. 6 addresses this question and suggests that CC based selection can indeed improve other recombination heuristics, but not all. The experiment tested six different heuristic conditions, i.e.: (1) PM, (2) one-point CO, (3) simple DE, (4) PM + mutation, (5) CO + mutation and (6) DE + mutation. Each one of the heuristic conditions was tested on data 7 (depicted in the graph) with a target error of 0.35, and using three different types of solution selection methods (i.e., RS, *ccMat3* and *ccMat6*). Each combination (heuristic and selection method) was tested 30 times.

The first obvious observation that can be made after a quick inspection of Fig. 6 is that CC based selection helps PM and CO, but not DE. In fact it even makes DE worse. Secondly, CO benefits from *ccMat6* but not from *ccMat3*. Thirdly, as we add heuristics (e.g., mutation) the relationships between heuristics and selection method, with regards to convergence speed,

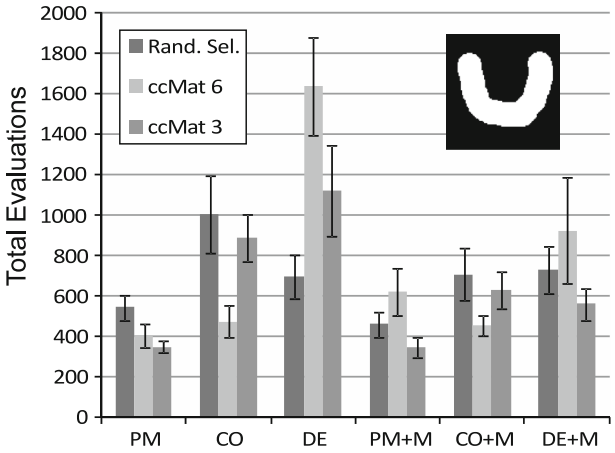

Fig. 6 The effect of CC based selection on different solution recombination heuristics

becomes more unpredictable (e.g., PM+M and *ccMat6*). Fourthly, when we add mutation to DE (i.e., DE+M), CC based selection (i.e., *ccMat3*) seems to help conversely to what we observe in the pure DE case. In conclusion, adding complementarity information to the solution selection process seems to be generally robust to the type of fusion heuristic adopted (e.g., PM and CO) but is not always helpful (e.g., DE). The latter case may be explained by the fact that complementarity increases the probability that the chosen solutions are excessively different from each other, making the resulting differentials (or velocity vectors) meaningless.

3.6 Experiment 6

Another important question was concerned with the temporal evolution of different measures under RS and CCS conditions. Figure 7 depicts the temporal evolution (the x-axis depicts the generation number) of four different measures: (7a) the error of the best solution within each generation, (7b) the mean error of all solutions within each generation, (7c) the best CC error of the best complementary pair in each generation and (7d) the number of solutions pairs in each generation with CC error smaller than 0.05 and with a minimum overlap of correct classifications of 1 (a minimum overlap of 1 implies $N^{11} \geq 1$ as seen for example in *ccMat3* in Table 4). For the experiment depicted in Fig. 7, data 1 was used and 20 tests were repeated for each condition.

Figure 7a again shows that CC based selection can significantly improve the convergence speed of neuroevolution. Perhaps, somewhat surprisingly, there doesn't appear to be a large difference between the solution selection conditions, in terms of the general trend of mean generational error, as shown in Fig. 7b, although the variance of this error appears to be significantly higher for the RS case. The evolution of CC error (Fig. 7c) for the best complementary solution pair in each generation is relatively similar (improving over time in both cases), but is relatively better for the CC condition from about generation 30 to 75. For Fig. 7d, the number of "good" complementary pairs initially grows in a similar way for both conditions (from generation 1 to about 30), then increases more rapidly for CCS (from about generation 30 to 75) and is finally dominated by RS (from about generation 75 to 200). In conclusion, CC based selection has again been shown to improve convergence speed (Fig. 7a), but its effect on different error related measures can be complex and unpredictable. Note

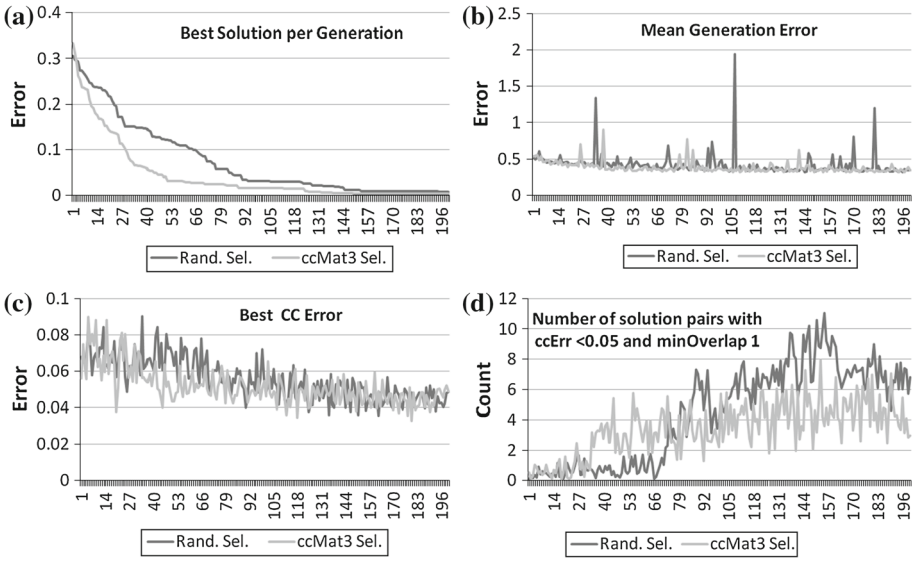

Fig. 7 The temporal evolution of different error related measures under different solution selection conditions

that in these experiments we are not directly manipulating any CC related measures, we are only using complementarity information in order to guide the selection of solution pairs, which has an indirect effect on CC related measures.

3.7 Experiment 7

Up to now the relative merits of CC based selection have been investigated under a constant set of NDM parameters. What happens if we vary an important structure-defining parameter namely, WF set cardinality? In most of the experiments thus far, seven distinct WFs (and three NFs) have been available for the specification of nodes in a network. What happens if at the start of each optimization experiment we fix this set of available WFs to a specific cardinality and select the constituent WFs randomly? Figure 8 depicts one such experiment. Both sub-figures illustrate different aspects of the same experiment on data 6, where the WF set cardinality was varied from 1 to 7, and where each combination of conditions was tested 50 times. In Fig. 8a, each WF set cardinality is associated with two selection conditions (i.e., random and complementarity based), and the y-axis depicts the mean error at generation 125. In Fig. 8b, the temporal evolution of the error for WF set cardinalities 1, 2, 6 and 7 is depicted, for the CCS condition.

Figure 8 is suggestive of the advantages of both NDMs and complementarity based selection. Both sub-figures show that convergence tends to improve with WF set cardinality. In other words, as the number of available WFs increases, so does the convergence speed of neuroevolution. Although the addition of functional diversity will tend to increase the number of local minima, this seems to be compensated by a corresponding increase in computational capacity, at least in this context of small networks (i.e., two input nodes, four middle nodes and one output node). Note also how for all WF set cardinalities except 6, CC based selection improves convergence relative to RS.

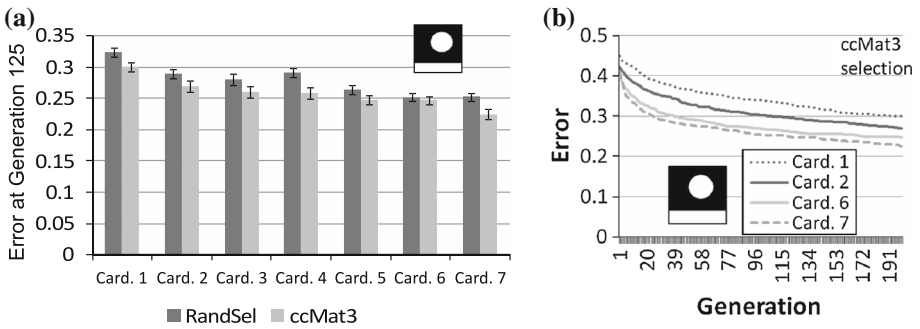

Fig. 8 The effect of WF set cardinality and solution selection method on convergence

3.8 Experiment 8

Figure 8 provides a strong motivation to look deeper into the influence of specific WFs or WF-subsets on the performance of NDMs. With this in mind, we ran a large experiment whereby every possible WF subset (out of 127 in total) was tested on data 6, and compared in terms of the RS and complementarity (*ccMat3*) selection conditions. For each WF subset, and both selection conditions, 40 tests were run, where each optimization run consisted of 150 generations. For example, if for a particular optimization run, the WF subset {1, 2} is selected, this means that the nodes of the evolving neural networks will only be allowed to use inner-products and Euclidean distances for their WFs (refer to Table 2 to recall the WF and NFs available).

This large experiment again confirmed the relative advantage of CC based selection and revealed several interesting facts about WFs and WF subsets, at least in the context of data 6. Firstly, if WF subsets are ranked in terms of their average error for the RS condition, it is found that the top 50 % WF subsets all contain the HO product WF. The top 50 % WF subsets for the CCS condition are exactly the same as those for the RS condition and therefore the same observation applies. This is a striking result which confirms earlier results demonstrating the usefulness of higher-order neural networks [16,25]. According to the experiment, the best WF subset for the RS condition was {2, 3, 4}, whereas for the CCS condition it was {2, 3}. The useful combination of Euclidean distance, logistic regression and HO product nodes has been observed before for example in the work of Gutiérrez et al. [16]. Secondly, if the errors for all WF subsets under the RS and CCS conditions are averaged the advantage of complementarity based selection is again evident, i.e.: 0.293 (RS) and 0.274 (CCS), where the means are significantly different ($t = 21.24$, $df = 126$, $P < 0.01$, paired t test). Thirdly, a weak positive correlation ($r = 0.204$) between the cardinality of WF subsets and the CC advantage (i.e., difference between the RS error for a particular WF subset and the corresponding CCS error) was found. Moreover, moderate negative correlations were found between WF subset cardinality and RS error ($r = -0.34$) and WF subset cardinality and CCS error ($r = -0.39$). This is consistent with previous observations (Fig. 8) that larger WF subset cardinalities tend to lead to lower errors.

Figure 9a again illustrates the striking advantages conferred by HO products. Each point in the scatter plot corresponds to a different WF subset. The x-axis coordinate for a particular point corresponds to the difference between the RS error and the CCS error for a particular WF subset, whereas the y-axis represents the corresponding average between these errors. The two different clusters correspond to those WF subsets that contain HO products and those that

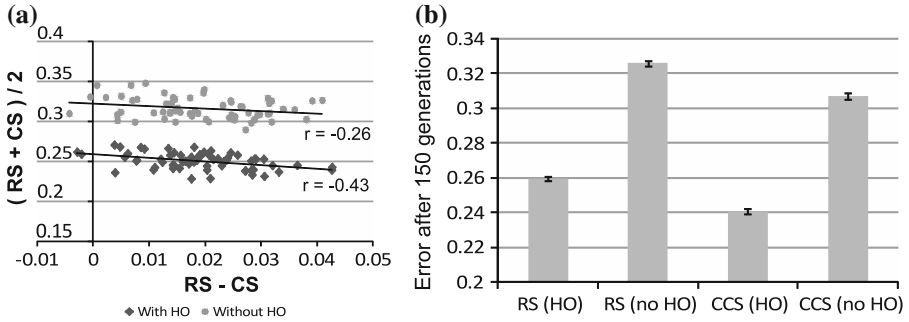

Fig. 9 The advantages of HO weight functions and complementarity based selection

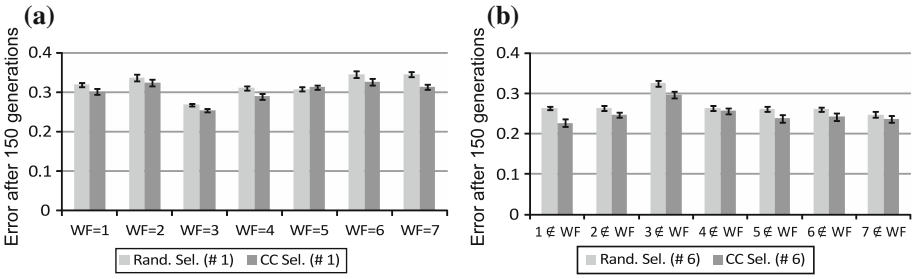

Fig. 10 Comparing WF set cardinalities *1* and *6*

do not. Notice the strong negative correlation between "CCS advantage" and "average error for a particular WF subset" in the HO cluster. As the average error decreases (i.e., as the WF subset improves), the CCS advantage increases. Figure 9b clearly shows both the advantage of having HO products in WF subsets, and the advantage of using CC based selection. The lowest average error corresponds to those WF subsets that contain HO products and those cases where CCS is used.

Figure 10 depicts the relative strengths of WF subsets with cardinality 1 (Fig. 10a) and cardinality 6 (where different single WFs have been dropped; Fig. 10b). These sub-figures again demonstrate the usefulness of the HO product WF, i.e.: it exhibits the lowest error for the cardinality 1 condition and if absent in the cardinality 6 condition, this leads to the highest error. The CCS advantage is noticeable for the majority of cases depicted in Fig. 10. According to Fig. 10b, in the CCS condition, the inner-product WF appears to be the most dispensable WF. In the cardinality 1 case, the least useful WFs seem to be the minimum and maximum WFs.

Table 5 depicts the detailed errors for WF subsets of cardinality 2, ordered by increasing $(RS+CCS)/2$ errors. The means for the RS and CCS conditions, 0.31 and 0.29, respectively, are again significantly different ($t = 6.2$, $df = 20$, $P < 0.01$, paired t test). Notice how the top six WF subsets all contain HO product WFs and the next five WF subsets contain HO subtractive WFs. Thus, the top 11 WF subsets of cardinality two, all contain at least one type of higher-order WF.

3.9 Experiment 9

In order to verify that the complementarity effect was not restricted to small two-dimensional data-sets we conducted a few sanity tests on several larger real-world datasets. The comple-

Table 5 WF sets of cardinality 2

Ranks	WF sets	Error after 150 generations Rand.Sel.	CC Sel.	(RS + CCS)/2
1	{2, 3}	0.246	0.216	0.231
2	{3, 4}	0.251	0.224	0.238
3	{3, 5}	0.245	0.234	0.239
4	{1, 3}	0.265	0.250	0.257
5	{3, 7}	0.272	0.263	0.267
6	{3, 6}	0.270	0.266	0.268
7	{2, 4}	0.312	0.284	0.298
8	{4, 7}	0.303	0.295	0.299
9	{1, 4}	0.306	0.292	0.299
10	{4, 6}	0.309	0.296	0.303
11	{4, 5}	0.314	0.309	0.311
12	{2, 5}	0.320	0.306	0.313
13	{1, 7}	0.332	0.297	0.314
14	{1, 6}	0.332	0.305	0.319
15	{1, 5}	0.331	0.308	0.320
16	{1, 2}	0.338	0.307	0.322
17	{5, 7}	0.332	0.324	0.328
18	{6, 7}	0.330	0.331	0.331
19	{5, 6}	0.334	0.329	0.331
20	{2, 7}	0.345	0.345	0.345
21	{2, 6}	0.349	0.342	0.345

Table 6 CCS advantage on several data benchmarks

Datasets	Samples	Architectures	Generations	RS error	CCS error	P(T ≤ t) one-tail
Iris	120	4–4–3	30	0.59	0.53	0.00716
Heart 1	214	22–1–1	20	0.069	0.066	0.02952
Abalone	835	8–2–1	20	4.749	3.721	0.00066

Number of tests per condition 30, CC matrix type ccMat11

mentarity advantage was found to be statistically significant for versions of the Iris, Heart and Abalone datasets (refer to Table 6). It must also be pointed out that although complementarity based selection was never worse than random based selection, the former was not always found to be statistically significantly better. For example, several different complementarity versions (i.e., *ccMat* 3, 6, 7 and 11) were tested on the Diabetes training dataset from the Proben1 set of problems [29], adopting 8–2–1 network architectures, without any statistically significant differences between random and complementarity based selection.

4 Discussion

4.1 Experiments

The results above can be summarized into the following main points:

- PM is an effective recombination heuristic (Fig. 2),
- complementary solutions can be effectively recombined (using deep recombination) (Fig. 3),
- complementarity emerges naturally even with simple random initialization (Fig. 4),
- complementarity can indeed be beneficially integrated into the solution selection process (for recombination) (Fig. 5),
- this integration can be done vis-a-vis different recombination heuristics (Fig. 6) and
- the advantage conferred by complementarity-based selection is robust to different data-sets (Table 6 and other results).

In the following subsections we delve a bit deeper into the implications of the study's several experiments.

4.1.1 Experiments 2 and 3

The aim of the experiment depicted in Fig. 3 was to confirm the viability of using CC information in order to improve neuroevolutionary convergence. The experiment succeeded in showing the potential of this information thus justifying subsequent experimentation. The fact that deep recombination errors were worse than corresponding "CC errors" (and yet much better than the best individual error from the solution pair) suggests that "deep recombination" strategies deserve further attention. Assuming that it is possible to reliably identify useful complementary pairs, better deep recombination strategies might be capable of leading to faster convergence and networks that generalize better. In Fig. 4 it can be seen that complementarity emerges easily (just by adding new random solutions to a population), however there is still room for improvement. The experiment embodied in Fig. 4 can be extended to introduce and compare different complementarity-promoting mechanisms. What constitutes "useful complementarity", especially in the context of generalization, would also require further investigation. Two complementary solutions, each one of which divides the classification space in a simple but distinct way, are more likely to be useful, than two equally complementary solutions, which however exhibit overly complex classification boundaries.

4.1.2 Experiment 4

The experiment depicted in Fig. 5 provides clear evidence for the benefits accrued from allowing complementarity information to influence the selection of solutions for recombination. When solution pairs with lower complementary error are more likely to be chosen for recombination, this leads to more effective convergence. All conditions show the CC based selection advantage. However, different variants of CC based selection seem to work best for different data-sets (e.g., data 6 seems to work best with *ccMat6* whereas data 7 seems to work best with *ccMat3*). This suggests that more research is required in order to find a single robust CC based selection method that is optimal for the broadest possible range of data-sets.

4.1.3 Experiment 5

The experiment depicted in Fig. 6 is indicative of the fact that CC based selection can be successfully applied to different recombination heuristics. So far, it has been shown that both PM and traditional CO benefit from CC based selection. It was also shown that the performance of pure DE deteriorates with CC based selection, most probably because differentials (or velocity vectors) resulting from excessively disparate solutions (due to complementarity) are meaningless. It was also shown that when mutation is added to DE, the CC advantage becomes manifest again (with *ccMat3*), presumably because mutation counteracts the initial deterioration caused by excessively disparate solutions. We hypothesize that the mutation heuristic accelerates optimization into a region where complementary solutions are no longer so disparate from each other and therefore complementary solutions no longer lead to meaningless differentials. It would be interesting to do an exhaustive study investigating which types of CC based selection methods work with which types of recombination heuristics. Moreover, it may even be possible to incorporate CC based information into particle swarm optimization and other population based optimization approaches.

4.1.4 Experiment 6

The results depicted in Fig. 7 again demonstrate the robustness of the CCS effect, particularly in Fig. 7a, where it is shown that the temporal evolution of the best error is always better for the CCS condition as compared to the RS condition. The results in Fig. 7c, d suggest that the temporal evolution of CC related measures (e.g., error of the best CC solution pair) is not always intuitive. However, due to the noisiness of the corresponding curves, which average the results of 20 tests, it is too early to determine unequivocally the true shape of the curves and therefore to draw any conclusions. Having said that, Fig. 7d in particular, seems to hint at a very interesting relationship where the number of "good" CC solutions is initially similar for both CCS and RS conditions, then larger for the CCS condition and finally larger for the RS condition. This leads us to tentatively suggest an optimization protocol (within the hyper-heuristic family of approaches) that varies the type of selection method adopted at different generations (e.g., random, *ccMat3* and *ccMat6*), based on different population statistics and/or cost dynamics.

4.1.5 Experiments 7 and 8

The results in Fig. 8 depict our first investigation into possible relationships between NDM factors and CC related conditions. In particular it shows that neuroevolutionary convergence is accelerated by both having larger sets of available WFs and using CC based selection. Figure 8a in particular shows that for practically every WF set cardinality, CC based selection confers an advantage. It is unlikely that by indefinitely increasing WF cardinality the advantages observed in sub-figures a and b in Fig. 8 will always hold. The relationship is likely to be complicated by a compromise between computational capacity and the density of local minima, which is also likely to depend on the size of the neural networks involved, the stage of evolution that is being considered and peculiarities of the problem (or dataset) being tackled. The subsequent results (e.g., Figs. 9, 10; Table 5) from Experiment 8, attempt to unravel these questions further. These questions are in great part motivated by model selection and the goal of developing fast algorithms for determining the appropriate WF subsets (e.g., higher-order product and Euclidean distance nodes) and the appropriate architectures (e.g.,

one feed-forward and one recurrent layer) for particular problems. The results again show the advantage of CC based selection and furthermore replicate earlier results by several authors demonstrating the usefulness of higher-products. Notice for example the "CCS (HO)" condition in Fig. 9b: enforcing the availability of HO WFs together with CCS confers the best convergence.

4.1.6 Experiment 9

Although the results of this final experiment were not an unequivocal success they did show us that the advantages of CCS can be found in more realistic and complex datasets. However, they were also a reminder of how the benefits of complementarity have only begun to be tapped and that more work is required to scale-up the approach to larger datasets.

4.2 Strengths

The complementarity based selection effect is significant and robust. The current paper has shown how complementarity information significantly improves convergence in a vast range of conditions. Therefore applications of complementarity in new contexts are likely to bear fruit with minimal fine-tuning.

The concept is simple and its application to new evolutionary algorithms is also likely to be simple. In this paper, we have applied it to the selection of solution pairs, via a ccMat, which requires very modest computational and memory resources.

The concept is also general. Although we applied CC information to solution selection, it is likely to be beneficial in other aspects of evolution, e.g.: initialization, generation trimming, and others. Furthermore, the concept of complementarity is not restricted to classification (or pattern recognition). There are many other optimization problems for which some useful notion of complementarity can be envisaged and which therefore might benefit from this approach. Multi-objective optimization problems are probably the most obvious candidates, where complementarity would be measured with respect to individual objectives (e.g., in a hypothetical three objective problem, solution 1 might perform well in objectives A and B, while solution 2 might perform well in objectives B and C, rendering this a good complementary solution pair).

4.3 Weaknesses

Probably the greatest weakness observed in the experiments so far pertains to the extent of improvement afforded by complementarity-based selection in its current form, which was lower than expected. It is important to note that although the improvement was not as large as expected it was indeed significant and relatively robust, i.e.: it passed the rigours of statistical tests and it was observed in many different conditions and datasets. Regarding robustness, as already mentioned in the Sect. 3, the complementarity advantage was not always detected, and therefore more work is required not only to strengthen the effect but also to broaden its applicability (e.g., larger and more complex data-sets). This research represents an early step towards deriving the benefits of complementarity, and it is hoped that future work will eventually uncover optimal complementarity based strategies for maximizing convergence speed and generalization capability.

Another weakness of the current work, already alluded to, pertains to the fact that different strategies (e.g., *ccMat3* and *ccMat6*) seem ideally suited to different data-sets. It is hoped that with more research, a single general complementarity-based strategy can be developed,

which is capable of tackling a majority of problems. If this is not possible, then a general approach for automatically switching between and/or selecting specific strategies should be aimed for.

4.4 Future Work

This work suggests several directions for future research, two of which are summarized here. Firstly, we would like to take a deeper look into CCS strategies themselves. Currently the method is based on extracting a subset of solution pairs with an acceptable level of complementarity, and then randomly selecting a pair from that subset. In the future it would be useful to investigate other approaches to the selection process, e.g.: pairs probabilistically selected in a manner directly proportional to their complementarity. Secondly, given the fact that evolutionary NDMs, in their current form, do not easily scale up to larger networks and larger datasets (in dimensionality and sample size), it would be useful to integrate evolutionary mechanisms, including CCS, with gradient-based mechanisms. Early work suggests that the approach is scalable in spite of significant neural diversity, but more work is needed to fully integrate the search paradigms. Future work is likely to benefit mostly from broadening the application of complementarity information and improving its impact on the convergence speed of optimization methods and the generalization capacity of machine learning algorithms.

References

1. Ackley DH (1987) A connectionist machine for genetic hillclimbing, vol 28. Kluwer Academic Publishers, Boston
2. Belew RK, McInerney J, Schraudolph NN (1991) Evolving networks: using the genetic algorithm with connectionist learning. Technical report CS90-174. Computer Science Engineering Department, University of California, San Diego (revised)
3. Cao J, Lin Z, Huang GB (2012) Self-adaptive evolutionary extreme learning machine. Neural Process Lett 36(3):285–305
4. Chicano F, Whitley D, Alba E (2014) Exact computation of the expectation surfaces for uniform crossover along with bit-flip mutation. Theor Comput Sci 545:76–93
5. Ding S, Li H, Su C, Yu J, Jin F (2013) Evolutionary artificial neural networks: a review. Artif Intell Rev 39(3):251–260
6. Duarte-Mermoud M, Beltrán N, Salah S (2013) Probabilistic adaptive crossover applied to Chilean wine classification. Math Probl Eng 2013:10. doi:10.1155/2013/734151
7. Duch W, Jankowski N (2001) Transfer functions: hidden possibilities for better neural networks. In: ESANN. Citeseer, Bruges, pp 81–94
8. Evans IK (1997) Enhancing recombination with the complementary surrogate genetic algorithm. In: IEEE international conference on evolutionary computation. IEEE, Indianapolis, pp 97–102
9. Fahlman SE, Lebiere C (1989) The cascade-correlation learning architecture. In: Touretzky DS, Hinton G, Sejnowski T (eds) Advances in neural information processing systems II. Morgan Kaufmann, San Mateo
10. Floreano D, Dürr P, Mattiussi C (2008) Neuroevolution: from architectures to learning. Evol Intell 1(1):47–62
11. Freire AL, Barreto GA (2014) A new model selection approach for the ELM network using metaheuristic optimization. In: ESANN 2014 proceedings, European symposium on artificial neural networks, computational intelligence and machine learning, pp 619–624
12. Gomez F, Miikkulainen R (1997) Incremental evolution of complex general behavior. Adapt Behav 5(3–4):317–342
13. Gomez F, Schmidhuber J, Miikkulainen R (2008) Accelerated neural evolution through cooperatively coevolved synapses. J Mach Learn Res 9:937–965
14. Gutiérrez PA, Hervás-Martínez C (2011) Hybrid artificial neural networks: models, algorithms and data. In: Advances in computational intelligence. Springer, Berlin,pp 177–184

15. Gutiérrez PA, Hervás-Martínez C, Carbonero M, Fernández JC (2009) Combined projection and kernel basis functions for classification in evolutionary neural networks. Neurocomputing 72(13):2731–2742
16. Gutiérrez P, Segovia-Vargas M, Salcedo-Sanz S, Hervás-Martínez C, Sanchis A, Portilla-Figueras J, Fernández-Navarro F (2010) Hybridizing logistic regression with product unit and RBF networks for accurate detection and prediction of banking crises. Omega 38(5):333–344
17. Hancock PJB (1992) Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. In: International workshop on combinations of genetic algorithms and neural networks, 1992, COGANN-92. IEEE, pp 108–122
18. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
19. Howard G, Bull L, de Lacy Costello B, Gale E, Adamatzky A (2014) Evolving spiking networks with variable resistive memories. Evol Comput 22(1):79–103
20. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501
21. Jaeger H, Haas H (2004) Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science 304(5667):78–80
22. Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach Learn 51(2):181–207
23. Masland RH (2001) Neuronal diversity in the retina. Curr Opin Neurobiol 11(4):431–436
24. Maul T (2013) Early experiments with neural diversity machines. Neurocomputing 113:36–48
25. Maul T, Baba S (2011) Unsupervised learning in second-order neural networks for motion analysis. Neurocomputing 74(6):884–895
26. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5(4):115–133
27. Miikkulainen R (2014) Evolving neural networks. In: Proceedings of the 2014 conference companion on genetic and evolutionary computation companion. ACM, New York,pp 487–512
28. Moriarty DE (1997) Symbiotic evolution of neural networks in sequential decision tasks. PhD Thesis, University of Texas at Austin
29. Prechelt L (1994) Proben1a set of neural network benchmark problems and benchmarking rules. Technical report 21/94. Fakultät für Informatik, University of Karlsruhe, Karlsruhe
30. Rempis C, Pasemann F (2012) An interactively constrained neuro-evolution approach for behavior control of complex robots. In: Variants of evolutionary algorithms for real-world applications. Springer, Berlin, pp 305–341
31. Schaffer JD, Whitley D, Eshelman LJ (1992) Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: International workshop on combinations of genetic algorithms and neural networks, 1992, COGANN-92. IEEE, Baltimore, pp 1–37
32. Schapire RE, Freund Y (2012) Boosting: foundations and algorithms. MIT Press, Cambridge
33. Semenkin E, Semenkina M (2012) Self-configuring genetic programming algorithm with modified uniform crossover. In: 2012 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–6
34. Soltesz I (2006) Diversity in the neuronal machine: order and variability in interneuronal microcircuits. Oxford University Press, New York
35. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10(2):99–127
36. Stanley KO, D'Ambrosio DB, Gauci J (2009) A hypercube-based encoding for evolving large-scale neural networks. Artif Life 15(2):185–212
37. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
38. Whitley D, Starkweather T, Bogart C (1990) Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Comput 14(3):347–361
39. Xiao J, Zhou J, Li C, Xiao H, Zhang W, Zhu W (2013) Multi-fault classification based on the two-stage evolutionary extreme learning machine and improved artificial bee colony algorithm. Proc Inst Mech Eng C 228:1797–1807
40. Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447
41. Zhu QY, Qin AK, Suganthan PN, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recognit 38(10):1759–1763