Convergence and Crossover: The Permutation Problem Revisited

Tom Froese and Emmet Spier

CSRP 596

June 2008

ISSN 1350-3162

Cognitive Science
Research Papers

Convergence and crossover: The permutation problem revisited

Tom Froese and Emmet Spier

Centre for Computational Neuroscience and Robotics (CCNR)
Centre for Research in Cognitive Science (COGS)
Âé¶¹Ó³»-, Brighton BN1 9QH, UK

t.froese@sussex.ac.uk

Abstract

Standard crossover operators are often omitted from simple genetic algorithms (GAs) used for optimizing artificial neural networks because of the traditional belief that they generally disrupt the distributed functionality of the evolving solutions. The notion that crossover will be especially disruptive when a genetic representation is used which has a many-to-one mapping between genotype and phenotype has become known as the 'permutation problem'. In contrast, this paper argues that these problems do not normally appear in practical use of simple GAs because populations converge quickly and then continue to move through search space in this converged manner until a fitness optimum is found. After convergence all individuals are genetically similar, and moreover, distinct genetic permutations of the same phenotypic solution are unlikely to co-exist in the population. Genetic convergence thus minimizes the possibility for disruption caused by crossover. We have termed this the 'convergence argument'. This claim is investigated experimentally on standard benchmark problems and the results provide empirical support.

Keywords: permutation problem, genetic algorithm, artificial neural network

1. Introduction

The optimization of artificial neural networks (ANNs) by using genetic algorithms (GAs) has matured and become an established discipline. Indeed, there are many successful applications of GAs using standard crossover operators to the optimization of ANNs. Nevertheless, it has often been claimed in the literature that the use of such operators can have detrimental effects on the evolutionary process. This is a concern that was particularly prevalent in the early 1990s (for an overview see Yao, 1999), but which has also been commented on more recently (e.g. García-Pedrajas, Ortiz-Boyer & Hervás-Martínez, 2006; Stanley & Miikkulainen, 2002). The source of this belief can generally be traced to a combination of two factors, namely (i) the observation that ANNs store their knowledge in a distributed fashion, and (ii) false assumptions about the nature of genetic convergence in evolutionary search (cf. Harvey & Thompson, 1996).

It is a well known fact that the functionality of a standard ANN is distributed over all of its nodes, connections and their weights (Yao, 1999). It is therefore possible that different arrangements of weights and connectivity can produce the same kind of overall behavior. Moreover, in standard ANNs the same set of nodes can be arranged in a variety of equivalent permutations since the order of nodes has no effect on the functionality of the corresponding ANN (Whitley, 1995). It has also been shown that the number of possible equivalent architectures grows exponentially with the number of hidden nodes (Schaffer, Whitley & Eshelman, 1992).The genetic redundancy caused by this many-to-one mapping from genotype to phenotype means that two functionally equivalent ANNs can have very different genetic representations as shown in Fig. 1.

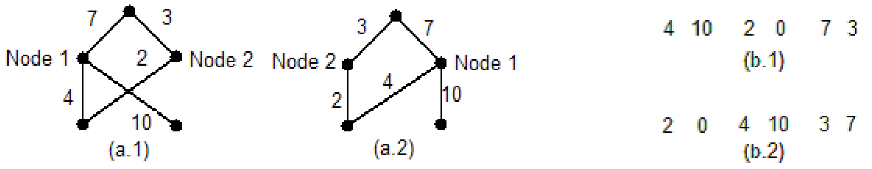


Figure 1. (a) Two functionally equivalent ANNs; (b) their differing genetic representation. Each weight is represented by an integer where 0 implies no connection (fig. adapted from Yao and Liu, 1997).

The *permutation problem* (Radcliffe, 1990) holds that evolved structures can get seriously disrupted by applying standard crossover between the two ANNs shown in Fig.1 because their genotypes are incompatible even though their phenotypes might be indistinguishable by the fitness function. This issue is also sometimes known as the *competing conventions problem* (Schaffer, Whitley & Eshelman, 1992), *structural-functional mapping problem* (Whitley, Starkweather & Bogart, 1990), and *isomorphism problem* (Hancock, 1992). An illustration of how the permutation problem may manifest itself is presented in Fig. 2.

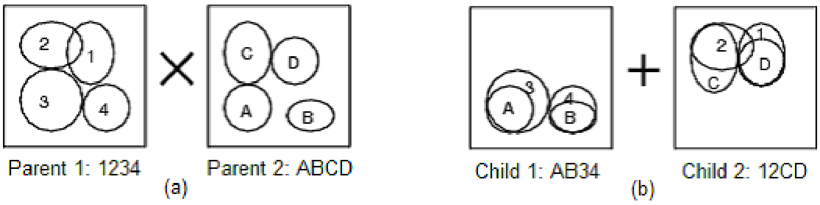


Figure 2. Recombination of ANNs with localized receptor fields: (a) parents selected for one-point crossover between the 2nd and 3rd node definitions, and (b) each offspring has two copies of similar receptor fields and does not cover the whole functionality of its parents (fig. adapted from Hancock, 1992).

The permutation problem is sometimes cited as a reason for using purely mutation based approaches (e.g. Yao, 1999; Yao & Liu, 1997), as well as a motivation for devising special genetic representations and/or crossover operators (cf. section 2.1). Indeed, it appears that “the prospect of evolving connectionist networks with crossover appears limited in general, and better results should be expected with reproduction heuristics that respect the uniqueness of the distributed representations” (Angeline, Saunders & Pollack, 1994). We note that these claims are based on theoretical arguments following the form of those presented in Fig. 1 and 2. Studies specifically investigating whether the permutation problem has any practical implications are relatively limited. This paper investigates ANN weight optimization, Hancock (1992) studied structural optimization, and García-Pedrajas, Ortiz-Boyer and Hervás-Martínez (2006) have recently investigated a combination of structure and weight evolution. It is worth emphasizing that none of these experimental studies has found any significant detrimental effects attributable to the permutation problem. Accordingly, there is a need to reevaluate the traditional theoretical claims with regard to this problem.

At first sight this lack of empirical evidence may seem odd, in particular because the permutation problem appears to encapsulate the reasonable claim that it usually makes little sense to recombine individuals who are genetically very dissimilar. As Watson and Pollack (2000) point out: “parents selected from two different fitness peaks are likely to produce an offspring that lands in the valley in between”. While we agree with Watson and Pollack’s observation, we further note that this situation is unlikely to occur in the case of real world applications of standard GAs. This is because it is improbable for several different fitness peaks, or permutations of the same fitness peak, to co-exist in the same population during evolutionary search since populations converge quickly onto a small region of genotypic search space. Indeed, it has been shown that even in the absence of selective pressure genetic convergence typically occurs during the initial generations through random genetic drift (Asoh & Muehlenbein, 1994). However, it is important to note that this genetic convergence does not necessarily entail premature convergence; in the case of real world applications the population usually continues to explore the search space in this converged manner until a stable fitness peak is found. In such cases the evolutionary path of the converged population through the space of possible genotypes will generally consist of phases of relatively directed movement up fitness slopes as well as random genetic drift along (fitness) neutral networks (Harvey & Thompson, 1996; Smith *et al.*, 2002; Ebner *et al.*, 2001). We propose that the widespread concern with the permutation problem in the literature stems from a disregard of the generally converged nature of practical GA-based search.

The purpose of this paper is therefore twofold: (i) to introduce the *convergence argument* to explain why standard crossover operators need not be harmful when used with simple GAs in a practical context (section 2), and (ii) to provide a series of experiments to obtain an indication of the extent of the permutation problem when using a simple GA with crossover to optimize ANN weights for classification tasks on two standard benchmark problems (section 3). The results of this series of experiments give empirical support to the convergence argument (section 4).

2. The permutation problem

In this section we review some of the work which has been done in order to address the permutation problem (section 2.1). We also introduce the convergence argument in order to explain why the problem does not typically appear when standard GAs are applied in a practical context (section 2.2).

2.1 Previous work

The various approaches of avoiding the permutation problem can be broadly grouped into two non-exclusive classes: (i) those that focus on improving the crossover operator, and (ii) those that focus on improving the genetic representation. The general aim is to adjust the overall crossover procedure in such a way that it is less likely to disrupt any distributed knowledge stored in the genetic representation.

It has been proposed that if one is somehow able to identify functional aspects of hidden nodes during the recombination procedure then this would allow the implementation of some form of “intelligent” crossover (Montana & Davis, 1989; see also García-Pedrajas, Ortiz-Boyer & Hervás-Martínez, 2006). One popular way of achieving this is to treat a node with its associated weights as one functional unit (e.g. Thierens, Suykens, Vandewalle & Moor 1993; Belew, McInerney & Schraudolph, 1992). Another suggestion is to reduce the adverse effects by placing incoming and outgoing weights of a hidden node next to each other in the genotypic representation. If the genotype is arranged in this manner it is possible to bias the crossover operator so that it is more likely to break the genotype at less disruptive points, e.g. between one node’s weight and another’s (e.g. Schaffer & Morishima, 1987).

However, this class of approaches faces three kinds of concerns: (i) on a theoretical level the attempt to localize ANN functionality for more targeted crossover appears counterintuitive when considering the distributed nature of standard ANNs, (ii) on a practical level it has been noted that designing such “intelligent” crossover operators could more than rival the complexity of the original learning problem (cf. Angeline, Saunders & Pollack, 1994), and (iii) on an experimental level it has been observed that in many cases simple crossover works better than the more sophisticated recombination algorithms (e.g. Hancock, 1992). A more promising approach might be to group genes for crossover using historical markers (e.g. Stanley & Miikkulainen, 2002).

The other class of approaches attempts to deal with the permutation problem by adjusting the genetic representation. Generally, the aim is to implement a one-to-one mapping between ANN architecture (genotype) and functionality (phenotype) so that several genetic permutations of the same phenotypic solution cannot co-exist in the same population. This can take the form of a special encoding mechanism that makes the order of nodes in the genetic representation irrelevant (e.g. Thierens, 1996; Radcliffe, 1993). However, it is important to emphasize that removing the possibility of genotypic permutations in this manner can be quite counterproductive in many cases. During his investigation of the permutation problem, Hancock (1992) observed to his surprise that this consistently produced worse results. Indeed, in contrast to the traditional view that a many-to-one mapping is an undesirable source of deception, it has recently been shown that the neutral search space afforded by the use of such a mapping function has the potential of significantly aiding the evolvability of a system (e.g. Shipman, Shackleton & Harvey, 2000; Harvey & Thompson, 1996). The neutral theory of evolution as genetic change without selection pressure was first introduced in biology by Kimura (1983); it has recently been the focus of increased interest in evolutionary computation and related fields (e.g. Smith *et al.*, 2002; Ebner *et al.*, 2001; Barnett, 2001; Izquierdo-Torres, 2004).

This brief overview of the relevant literature indicates the amount of effort which has been invested toward overcoming the permutation problem. It has also been pointed out that this work is faced by a number of theoretical and practical concerns. More importantly, previous experimental investigations into the actual practical severity of the permutation problem have revealed that in many cases the problem is not as severe as normally assumed (e.g. Hancock, 1992; García-Pedrajas, Ortiz-Boyer & Hervás-Martínez, 2006), a finding which is further supported by the results presented in this paper (section 4). What can account for this discrepancy between theory and practice?

2.2 The convergence argument

We introduce the *convergence argument* to explain why the possibility for disruption by the use of standard crossover operators is often insignificant when using a simple GA: (i) genetic convergence occurs during the initial generations after which most members of the population will have similar genetic representations, and therefore (ii) several significantly distinct permutations of the same solution are unlikely to co-exist. We agree with Harvey (1992) that in biological terms we could say that a simple GA typically adapts a particular converged population, or *species*. In this case using a standard crossover operator is likely to produce offspring with similar fitness to their parents.

This is a general argument that applies whenever there is a many-to-one mapping between genotype (which could be binary, real-valued, etc.) and phenotype (which could be ANN weights, structure, etc.); the permutation problem in the artificial evolution of neural networks weights is one well known example. The claim that using standard crossover in combination with such genetic representations tends to produce unfit offspring, as exemplified by the literature on the permutation problem, seems to result from a disregard of the generally converged nature of standard population-based search. The convergence argument therefore qualifies the common generalization that “it is generally very difficult to apply crossover operators in evolving connection weights since they tend to destroy feature detectors found during the evolutionary process” (Yao, 1999). In contrast, we note that standard crossover is usually not harmful in practice because for most of the generations of an evolutionary run the population will have converged onto one area of the genotypic search space which it continues to explore.

The convergence argument is supported by the observation that the two previous empirical studies which investigated the practical severity of the permutation problem, and which did not find any significant empirical evidence for its existence, made no use of any diversity preserving mechanism (Hancock, 1992; García-Pedrajas, Ortiz-Boyer & Hervás-Martínez, 2006). The GAs used in their experiments were therefore in all likelihood applying crossover to members of a converged population. Indeed, Hancock (1992) notes that “resolving the permutations is aided by high selection pressure: by increasing the dominance of the top-ranked string, it is better able to enforce its order on the population”.

The reason why a GA’s population generally converges so rapidly is that the selection operator reduces the genetic diversity of the population towards zero because a few fit individuals will quickly spread their genes throughout the population. In the presence of mutation the genetic diversity after convergence is not zero, but a higher balance between selection/genetic drift and mutation (Harvey & Thompson, 1996). It has been argued that it is mainly through mutation that fitter phenotypic solutions can be found even after genetic convergence; either by hill climbing or through genetic drift on a (nearly) neutral fitness landscape leading to punctuated increases in fitness (e.g. Harvey, 2001; van Nimwegen & Crutchfield, 2000; Barnett, 2001). In other words, there are two important factors at work here: (i) genetic convergence makes it unlikely that significantly different genetic permutations of the same phenotypic solution will co-exist in the population, thereby minimizing the possibility of disruption through crossover, and (ii) the fact that slightly different permutations may co-exist can improve evolvability, because it allows evolutionary search to traverse neutral networks toward higher fitness peaks.

According to the convergence argument we can expect the crossover operator to be more disruptive in evolutionary runs where the population has a longer time to convergence, for example when using very large population sizes or when diversity preservation methods such as niching are used (e.g. Stanley & Miikkulainen, 2002). In such cases it might be more appropriate to make use of one or more of the methods mentioned in section 2.1 in order to minimize the possibility of disrupting any evolved solutions, in particular when applying crossover to individuals which have been selected for recombination from different niches. Moreover, we can expect the crossover operator to generally work better with smaller populations. This intuition is supported by Belew, McInerney and Schraudolph (1992) who report that because an ANN’s configuration is “*undetermined* by the problem it is trying to solve,” its various permutations are unlikely to share the same schemata and thereby make the GA less effective. It is suggested that keeping the population size small will reduce the disruption caused by competing permutations because, “if very small populations are used with the GA, there is not ‘room’ for multiple alternatives to develop”. We agree, while further noting that in small populations there is not enough ‘room’ for multiple alternatives (which might be present in the population initially) to *persist* in the face of selection pressure which forces convergence on one particular fitness peak.

In order to obtain more insight into the validity of the convergence argument we ran a series of experiments on two standard benchmark problems. The study focused on the effects of standard crossover on the ANN classification accuracy and GA efficiency.

3. Experiments

3.1 Experimental data

The effect of standard crossover on the artificial evolution of ANNs was investigated by applying this technique to two real problems in the medical domain, namely breast cancer and diabetes diagnosis taken from the “Proben1” benchmark set (Prechelt, 1994). A practical advantage of choosing these datasets is that they have already been used in research on crossover and the permutation problem by García-Pedrajas, Ortiz-Boyer and Hervás-Martínez (2006), and that ANNs are among the most common methods for breast cancer diagnosis (Abass, 2002). We also agree with Prechelt (1994) that results obtained on real world data will be more revealing than if the ANNs were trained on an artificial task, in particular because we are interested in whether the permutation problem manifests in practice. It has been suggested that whereas it is possible to devise special fitness landscapes with isolated hills such that genetic convergence is likely to coincide with premature convergence on a local fitness optimum, fitness landscapes associated with many real problems are not of this nature (Harvey & Thompson, 1996).

The breast cancer database was originally obtained from the University of Wisconsin Hospitals in Madison by Dr. W.H. Wolberg (Wolberg & Mangasarian, 1990). This dataset was chosen because it has been used widely in the literature (e.g. Abass, 2002; Xao & Liu, 1997; Fogel, Wasson & Boughton, 1995; Prechelt, 1994), and is still in use today (e.g. García-Pedrajas, Ortiz-Boyer & Hervás-Martínez, 2006; Ortiz-Boyer, Hervás-Martínez & García-Pedrajas, 2005). It also represents one of the easier Proben1 benchmark sets; this is important because it is very likely that many equivalent solutions exist, which should thus theoretically magnify the adverse effects associated with the permutation problem. For this dataset the ANNs were required to discriminate between benign and malignant tumors based on 9 different factors. There are a total of 699 samples in this dataset with 65.5% of the examples being classified as benign. In order for the classification results to be comparable with results presented in the literature (e.g. Fogel, Wasson & Boughton, 1995), the 16 records with missing values were removed from the dataset and no validation set was used. The first 400 records were then chosen as the training data while the remaining 283 records constituted the testing data.

The diabetes data set was created by Vincent Sigillito from John Hopkins University from a larger database held by the National Institute of Diabetes and Digestive and Kidney Diseases. This dataset has also been extensively investigated in the literature (e.g. Yao & Liu, 1997; Ortiz-Boyer, Hervás-Martínez & García-Pedrajas, 2005; Prechelt, 1994; García-Pedrajas, Ortiz-Boyer & Hervás-Martínez, 2006) and has been chosen because it represents one of the more difficult cases of the Proben1 benchmark set. The classification is made on 8 different inputs. There are a total of 768 samples available of which 65.1% are diabetes negative. No validation set was used in order to make the results comparable with those of the experiments using breast cancer data. The testing data is made up of 192 records while the training data consists of a total of 576 records by combining the training and validation data of the diabetes1 dataset. This combination is proposed by Prechelt (1994) for experiments that do not make use of a validation procedure. He also points out that the documentation provided with this dataset claims that there are no missing values; however, there are several senseless 0 values which most probably indicate missing data. We follow Prechelt and nevertheless treat these samples as real thereby probably introducing some additional noise into the dataset.

3.2 Experimental setup and implementation

The ANNs used in the classification experiments were basic feed-forward multi-layer perceptrons (MLPs) because these have been identified by Yao (1999) to likely increase the harmful effects of traditional crossover due to their distributed representation, and since they are still used in the context of breast cancer diagnosis (e.g. Abass, 2002). The activation function used was the standard sigmoid (logistic) function. Each node has a bias term associated with it. The ANN architecture used to classify the breast cancer data had 9 inputs, 9 hidden nodes and 1 output node (again following Fogel, Wasson and Boughton, 1995) for a total of 100 weights. The architecture used for the diabetes data had 8 inputs, two layers with 9 hidden nodes each and 1 output node for a total of 181 weights. The architectures were fully interconnected so that each node of every layer was connected with every node of the immediately following layer. The architectures were genetically represented by a list of floating-point numbers with the length of the list being equal to the number of connection weights of the encoded network architecture. The range of the weights was limited to the single precision C++ floating-point range.

The weights were initialized by drawing random numbers from the standard Gaussian distribution. The mutation operator was implemented as a probabilistic change of each connection weight by a small random floating-point number drawn from the standard Gaussian distribution. Three traditional kinds of standard crossover operators (uniform, one-point and two-point) were tested in a variety of GA settings[1]. While there are certain crossover operators which are more effective when used in combination with real-valued genotypic representations (such as directional crossover) these would generally decrease the number of generations required until population convergence. As such the traditional crossover operators chosen for our experiments represent a conservative choice with regard to the convergence argument. The fitness function used by the GAs evaluates an

[1] The GA software for this work was based on the GAlib package, written by Matthew Wall at MIT.

individual by testing the decoded ANN on the given training set. The percentage accuracy achieved on the classification task is then used as that individual's fitness. No scaling was applied to these scores. The selection mechanism used in all test runs was the popular roulette-wheel selection method, where each individual gets picked for mating in a probabilistic manner which is proportional to the individual's fitness score.

For most experiments the population size was set to 50 individuals and run for 1000 generations; when a population size of 500 was used, a typically large size (e.g. Fogel, Wasson & Boughton, 1995), the number of generations was reduced to 100 in order to make the two settings comparable with regard to computational cost. All experiments were conducted with two different types of GA. One variation was a steady-state GA that uses overlapping populations, and it was arbitrarily decided that the fittest 25% of the population overlaps between generations. A microbial GA (Harvey, 1996) was also tested as an example of a very minimal GA. The microbial GA uses a modified form of tournament selection in which two random members of the population get selected, an offspring is generated as usual by applying crossover and mutation, and is then used to replace the less fit parent. In each effective 'generation' this process is repeated as many times as there are individuals in the population. Since the fitter parent does not get modified during reproduction the microbial GA can be said to have 50% generational overlap on average.

Each of these simple GAs was tested with basic uniform, one-point and two-point crossover along with mutation, and also just with mutation alone as the control case. The probability of a particular gene getting mutated by adjusting it with a value drawn from the standard Gaussian distribution was set to 1%. The steady-state GA was additionally tested with a higher mutation probability of 2.5% to get an indication of whether the mutation rate has any effects on the permutation problem. In order to compare the impact of crossover in relation to the probability of its application during the generation of new offspring, each crossover operator was tested with two different probabilities (10% and 60%), and hence crossover was not used in the generation of every new individual. The microbial GA, which depends on a high crossover probability to implement selection, was extended to deal with cases where no crossover has taken place; when this happens the offspring is generated by mutating the fitter parent and replacing the less fit one. Note that this creates a strong selection pressure as less fit individuals are completely removed from the population, whereas they are only modified whenever crossover is applied.

All the combinations of settings described above were tested on both the breast cancer and diabetes datasets.

4. Results

The effect of crossover on the evolution of ANNs was analyzed from two different perspectives: (i) the generalization ability of the evolved solutions, namely the classification accuracy that the ANNs achieve on the testing data, and (ii) computational efficiency in terms of the number of evaluations required to evolve a particular weight configuration. This is a pragmatic choice since fitness evaluations are typically the most computationally expensive part of the evolutionary process. A summary of the classification accuracy results is presented first followed by a summary of the evaluations required to achieve those results. The following notation is used: m_p = probability of mutation, c_p = probability of crossover, uni. = uniform crossover, 1-p. = one-point crossover, 2-p. = two-point crossover, pop. size = population size.

4.1 Testing data classification accuracy

A summary of the breast cancer and diabetes testing data classification accuracy achieved by the evolved ANNs as a result of various GA settings can be found in Tables 1 and 2, respectively. To check if the use of crossover had any significant effect on the classification accuracy of the evolved ANNs the results of each GA variation with crossover were compared with those of the corresponding mutation-only variation using the 2-tailed student's *t*-test. All classification results are rounded to two decimal points.

The breast cancer results shown in Table 1a were not significantly different from each other with one exception where there was a statistically significant improvement in classification accuracy for a steady-state GA setting (1-p.; m_p = 2.5%; c_p = 60%; p_a < 0.0441) in comparison to its mutation-only variation. The percentage accuracies achieved for the breast cancer data with a population size of 500 are summarized in Table 1b; none of the settings produced results that were significantly different from using mutation alone. The diabetes classification results shown in Table 2 were not different from each other except for a microbial GA variation (1-p.; m_p = 1.0%; c_p = 60%; p_a < 0.023), which also showed a statistically significant increase in classification accuracy.

GA:	Uni. 10%	Uni. 60%	1-P. 10%	1-P. 60%	2-P. 10%	2-P. 60%	None
S-S 1.0%:	94.16	93.69	94.11	94.89	94.68	94.70	93.38
S-S 2.5%:	95.20	95.45	95.27	***95.78***[a]	95.52	95.10	94.53
Microbial:	94.65	94.77	94.58	94.96	94.98	94.96	94.63

Table 1a. Breast cancer *testing data*; classification accuracy averaged over 15 runs (pop. size = 50). One set of runs, highlighted in bold, was significantly better on average (p_a < 0.0441) than mutation alone. The highest classification accuracy is highlighted in italics.

GA:	Uni. 10%	Uni. 60%	1-P. 10%	1-P. 60%	2-P. 10%	2-P. 60%	None
S-S 1.0%:	95.59	95.74	94.79	95.03	95.08	95.31	95.22
S-S 2.5%:	95.74	*96.80*	95.92	96.14	96.09	95.88	96.21
Microbial:	94.44	94.70	95.01	94.94	94.70	95.22	94.63

Table 1b. Breast cancer *testing data*; classification accuracy averaged over 15 runs (pop. size = 500).None of the settings were significantly different from each other. The highest classification accuracy is highlighted in italics.

GA:	Uni. 10%	Uni. 60%	1-P. 10%	1-P. 60%	2-P. 10%	2-P. 60%	None
S-S 1.0%:	71.70	71.39	69.90	70.76	69.90	71.28	71.01
S-S 2.5%:	73.54	73.09	72.47	72.01	72.88	71.81	73.06
Microbial:	72.47	73.82	73.37	***75.49***[a]	73.30	74.69	73.47

Table 2. Diabetes *testing data*; classification accuracy averaged over 15 runs (pop. size = 50). One set of runs, highlighted in bold, was significantly better on average ($p_a < 0.023$) than mutation alone. The highest classification accuracy is highlighted in italics.

Note that the results presented here contrast the claims of the permutation problem. It seems that in these experiments the use of crossover generally made no difference to the evolved generalization ability. In addition, where there was a statistically significant difference in classification accuracy this was actually an improvement in generalization.

4.2 Number of evaluations

The number of evaluations of the fitness function provides a reliable measure of the computational cost incurred in finding a solution. It also allows an easier comparison of efficiency with other algorithms. A particular classification accuracy of the training data was selected, and the mean number of evaluations required to reach it was recorded. The training accuracy was chosen because it allows a simple assessment of the impact crossover has on search efficiency, and it avoids having to also evaluate the ANNs on the testing data at every generation. A classification target is said to be reached as soon as at least one of the individuals of a population achieves the required accuracy. For better comparison the targets were chosen so that most of the evolutionary runs would be able to satisfy the criteria. For the breast cancer data it was chosen to be 94%; for the diabetes data runs it was 78%. The number of evaluations that the breast cancer and diabetes experiments required to reach this target are summarized in Tables 3 and 4, respectively. All values are rounded to nearest integer.

GA:	Uni. 10%	Uni. 60%	1-P. 10%	1-P. 60%	2-P. 10%	2-P. 60%	None
S-S 1.0%:	3047	**1705**[a]	4049	**1890**[b]	3553	**1611**[c]	4186
S-S 2.5%:	2177	2259	2504	2363	2380	2496	3202
Micro:	3534	*1435*	2644	5193	2187	2690	2655

Table 3a. Breast cancer *training data*; evaluations taken for 94% accuracy averaged over 15 runs (pop. size = 50). Three settings, highlighted in bold, were significantly faster than mutation alone ($p_a < 0.0275$, $p_b < 0.0234$, $p_c < 0.0166$). The most efficient result is highlighted in italics.

GA:	Uni. 10%	Uni. 60%	1-P. 10%	1-P. 60%	2-P. 10%	2-P. 60%	None
S-S 1.0%:	987	**1219**[a]	880	**1315**[b]	1001	1071	930
S-S 2.5%:	1095	1286	992	1340	984	1283	1142
Micro:	738	**763**[c]	***422***[d]	811	472	591	589

Table 3b. Breast cancer *training data*; evaluations taken for 94% accuracy averaged over 15 runs (pop. size = 500). Four settings, highlighted in bold, were significantly different:

three were slower ($p_a < 0.0251$, $p_b < 0.0072$, $p_c < 0.0301$) and one was faster ($p_d < 0.0236$) than mutation alone. The most efficient result is highlighted in italics.

GA:	Uni. 10%	Uni. 60%	1-P. 10%	1-P. 60%	2-P. 10%	2-P. 60%	None
S-S 1.0%:	3091	4443	6027	3532	5631	4583	3289
S-S 2.5%:	2393	3028	2775	3546	4087	2686	3182
Micro:	4352	4267	*2164*	3279	4897	3238	3497

Table 4. Diabetes *training data*; evaluations taken for 78% accuracy averaged over 15 runs (pop. size = 50). The settings were not significantly different from each other for any of the GA variations. The most efficient result is highlighted in italics.

For the breast cancer data evaluations shown in Table 3a there were 3 crossover settings that were significantly different from mutation alone. The steady-state GA settings with m_p = 1.0% and using uniform (c_p = 60%; $p_a < 0.0275$), one-point (c_p = 60%; $p_b < 0.0234$) and two-point (c_p = 60%; $p_c < 0.0166$) crossover were all significantly more efficient in their search. Note that all three settings had a high crossover probability. This indicates that at least in the case of a steady-state GA with low mutation the effect of crossover might be to help evolutionary search. The corresponding runs with a higher probability of mutation (m_p = 2.5%) were not significantly more efficient than using mutation alone, which gives some evidence that the improvements found in the low mutation cases is due to an additional randomization provided by crossover. No support for the permutation problem was found.

Considering large populations, the number of evaluations required for the breast cancer experiments with a population size of 500 is given in Table 3b. There were two steady-state GA settings with m_p = 1.0% that fared significantly worse compared to using mutation alone, namely (uni.; c_p = 60%; $p_c < 0.0251$) and (1-p.; c_p = 60%; $p_d < 0.0072$). The microbial GA also had two settings which were significantly different from purely mutation based evolution. One crossover setting caused a reduction (uni.; c_p = 60%; $p_e < 0.0301$), and the other an improvement (1-p.; c_p = 10%; $p_f < 0.0236$) in search efficiency. All the crossover settings which were significantly less efficient had a high crossover probability (c_p = 60%). However, the different settings do eventually converge on the same solutions as there is no significant difference between the classification accuracies in Table 1b. Nevertheless, all settings produced runs that were significantly more efficient on average compared to their small population counterparts. This is likely to be the case because the breast cancer classification problem was relatively easy to solve, and thus not much fine-tuning was required to hit on a viable solution.

For the diabetes evaluations shown in Table 4, the steady-state GA settings with m_p = 1.0% produced 2.3 runs on average which did not make the final target and were thus removed when calculating the efficiency statistics. None of the GAs had any settings with crossover which took a significantly different number of evaluations compared to their mutation-only variations.

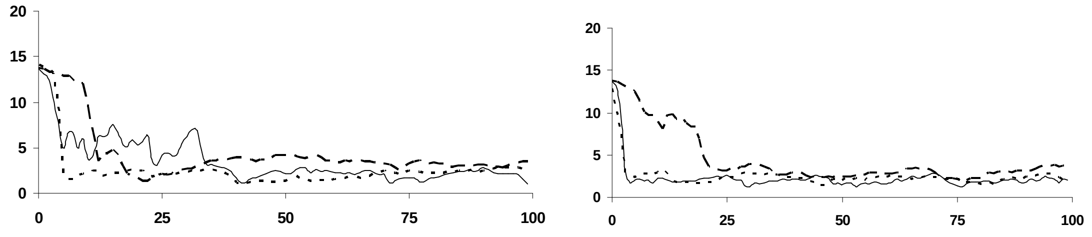
In general, no substantial support for effects attributable to the permutation problem was found, and using standard crossover operators mostly had little effect on the efficiency of the evolutionary search. Nevertheless, it is worth noting that Table 3a shows that the addition of crossover with c_p = 60% to a GA with a low mutation rate can make evolutionary search significantly more efficient. However, that advantage in efficiency is lost when the same settings are used in combination with a large population as shown in Table 3b. In these large population experiments the crossover operator is apparently more disruptive; an increased number of fitness evaluations are required for convergence on fit solutions compared to the runs which used mutation alone.

4.3 Population convergence

In the experiments conducted for this work the genetic diversity of the population was recorded after every generation. By plotting these records it is possible to show how the population converges over generations. For these experiments the genetic diversity was taken to be equal to the mean of the Euclidean distances between all the individuals' genetic encoding (considered as a real vector). The measure used for calculating the diversity d of a population (Pop) is shown in Eq. 1, where $\mathbf{p}$ is a 3 element, real-valued vector representing individual i's genotype.

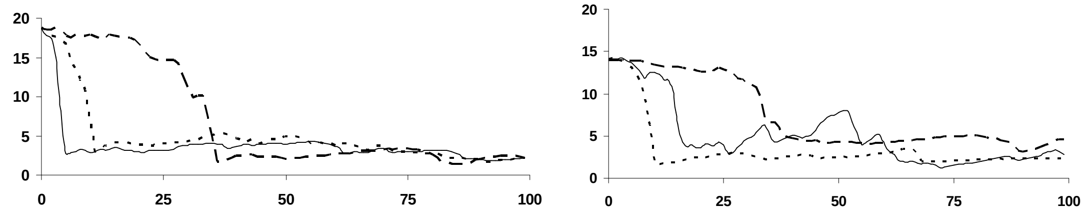
$$d = \frac{1}{\mathrm{Pop}^2} \sum_{i \in \mathrm{Pop}} \sum_{j \in \mathrm{Pop}} \left\| \mathbf{p}_i - \mathbf{p}_j \right\| \tag{1}$$

To illustrate how quickly a population generally converged in these experiments it will suffice to present a few examples here as shown in Fig. 3 and 4.



(a) (b)

Figure 3. Mean genetic diversity of representative runs plotted against the first 100 generations; breast cancer data (pop. size = 50): (a) steady-state GA; (b) microbial GA, both with m_p = 1.0%. *Legend:* long-dashes: uni. c.-o., c_p = 60%, short-dashes: uni. c.-o., c_p = 10%, and solid line: mutation alone.



(a) (b)

Figure 4. Mean genetic diversity of representative runs plotted against the first 100 generations: (a) settings as for Fig. 3a except using diabetes data; (b) settings as for Fig. 3a except with pop. size = 500. *Legend:* see Fig. 4.

Note that in all runs with a high probability of crossover the initial region of high genetic diversity is extended. This is likely due to the fact that the crossover operator can introduce new genetic variants when applied before genetic convergence. If using standard crossover does make a significant difference to the outcome and/or efficiency of an evolutionary run it could be because it has such an evident effect on the genetic diversity during the initial generations. Its impact will then decrease with convergence until it disappears when the genetic difference between individuals is limited to single mutational steps. Note also that, as expected, larger populations (500 individuals) with large amounts of crossover (c_p = 60%) require relatively more generations before genetic convergence, as illustrated in Fig. 4b when compared to Fig. 3a.

5. Discussion

The benchmark experiments presented in this paper provide two important results: (i) that a detrimental effect attributable to the permutation problem, a hypothetical problem often associated with the artificial evolution of neural networks when using standard crossover operators, was not found in most cases, and (ii) that crossover was generally applied to genetically converged populations. For most of the settings that we tested the effect of crossover is statistically negligible. In addition, our results show that in all experiments the use of a standard crossover operator never made the classification accuracy of the evolved solutions significantly worse than when using mutation alone. Further, in all experiments with a small population size (50) the inclusion of crossover was never found to increase the computational cost of the searches. However, we did find some support for an effect potentially attributable to the permutation problem in searches using a large population size (500) with a high probability (60%) of applying the crossover operator. In several of those experiments there was a statistically significant increase in computational cost compared to the runs with mutation alone; runs with large populations and a small probability (10%) of crossover were never found to have a statistically significant increase in computational cost. This supports the intuition that the crossover operator can potentially be more disruptive when used in conjunction with large populations, as discussed in section 2.2 of this paper.

We argue that the nature and degree of convergence of the populations (as illustrated in Fig. 3 and 4) provides a factor that can fully explain the summary of results above. A population that is fully converged will not experience any deleterious effect of crossing over ANNs because there are unlikely to be alternate permutations. Of course, normal evolutionary search will never be fully converged since there is a continual injection of new genetic material through mutation. If populations are converged to within the effects of the mutation operator then crossover will essentially become another (biased) mutation operator of similar magnitude and the permutation problem can not be manifest.

A disruptive effect attributable to the permutation problem might be present right at the beginning of the evolutionary search and during convergence. However, in some respects this is likely an additional population randomization contributing to a wider sampling of the search space. Certainly with small populations there is a real possibility that the initial population sampling will not contain members close to a global optimum; the additional randomization provided by the permutation effect of the crossover operator offers a mechanism to more fully sample the initial conditions of the search space. In the experimental results concerning small populations every significant difference in accuracy and efficiency of the crossover conditions compared to the pure mutation condition was found to be a beneficial effect attributable to the use of standard crossover, and in all these cases there was a large probability of crossover (60%).

6. Conclusion

What our work indicates is that the generalizations that have been published regarding the permutation problem cannot be justified and that it is more appropriate for the issue to be discussed in terms of population convergence and genotypic diversity. Accordingly, we proposed the ‘convergence argument’, namely that without the use of special diversity preserving mechanisms populations will typically converge quickly, and that after this convergence the use of standard crossover operators does not have an adverse effect. This is because at that point most genotypes will be very similar and it is unlikely that several distinct genetic permutations of the same phenotypic solution will be present in the population at the same time. The series of experiments on benchmark problems reported in this paper give empirical support to this convergence argument. We argue that the permutation problem does not generally appear in the practical application of artificial evolution to the optimization of ANNs because after a short transient post-initialization phase, the standard crossover operator is applied to a converged population.

Acknowledgements

The authors would like to thank Inman Harvey, Lionel Barnett, Nathaniel Virgo and Simon McGregor for their helpful comments and discussions.

References

Abass, H. A. (2002). An Evolutionary Artificial Neural Networks Approach for Breast Cancer Diagnosis. *Artificial Intelligence in Medicine*, *25*(3), 265-281.

Angeline, P. J., Saunders, G. M., & Pollack, J. B. (1994). An Evolutionary Algorithm that constructs Recurrent Neural Networks. *IEEE Trans. on Neural Networks*, *5*(1), 54-65.

Aso, H., & Muehlenbein, H. (1994). On the mean convergence time of evolutionary algorithms without selection and mutation. In H.-P. Schwefel, Y. Davidor, & R. Männer (Eds.), *Parallel Problem Solving from Nature III* (pp. 88-97), Berlin, Germany: Springer-Verlag.

Barnett, L. (2001). Netcrawling – Optimal Evolutionary Search with Neutral Networks. In J.-H. Kim, B.-T. Zhang, G. Fogel, & I. Kuscu (Eds.), *Proc. of the 2001 Congress on Evolutionary Computation* (pp. 30-37), Piscataway, NJ: IEEE Press.

Belew, R. K., McInerney, J., & Schraudolph, N. N. (1992). Evolving networks: Using the genetic algorithm with connectionist learning. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial Life II* (pp. 511-547), Redwood City, CA: Addison-Wesley.

Ebner, M., Langguth, P., Albert, J., Shackleton, M., & Shipman, R. (2001). On neutral networks and evolvability. In J.-H. Kim, B.-T. Zhang, G. Fogel, & I. Kuscu (Eds.), *Proc. of the 2001 Congress on Evolutionary Computation* (pp. 1-8), Piscataway, NJ: IEEE Press.

Fogel, D. B., Wasson, E. C., & Boughton, E. M. (1995). Evolving neural networks for detecting breast cancer. *Cancer Letters*, *96*(1), 49-53.

García-Pedrajas, N., Ortiz-Boyer, D., & Hervás-Martínez, C. (2006). An alternative approach for neural network evolution with a genetic algorithm: Crossover by combinatorial optimization. *Neural Networks*, *19*(4), 514-528.

Hancock, P. J. B. (1992). Genetic Algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. In D. L. Whitley, & J. D. Schaffer (Eds.), *Proc. Int. Workshop on Combinations of Genetic Algorithms and Neural Networks* (pp. 108-122), Los Alamitos, CA: IEEE Computer Society.

Harvey, I. (1992). Species Adaptation Genetic Algorithms: A Basis for a Continuing SAGA. In F. J. Varela, & P. Bourgine (Eds.), *Proc. of the 1st Euro. Conf. on Artificial Life* (pp. 346-354), Cambridge, MA: The MIT Press.

Harvey, I. (1996). *The Microbial Genetic Algorithm*. Retrieved Sept. 17, 2007, from the I. Harvey's Âé¶¹Ó³»-, Department of Informatics website: http://www.cogs.sussex.ac.uk/users/inmanh/

Harvey. I. (2001). Artificial Evolution: A Continuing SAGA. In T. Gomi (Ed.), *Evolutionary Robotics: From Intelligent Robots to Artificial Life* (pp. 94-109), Berlin, Germany: Springer-Verlag.

Harvey, I., & Thompson, A. (1996). Through the labyrinth evolution finds a way: A silicon ridge. In T. Higuchi, M. Iwata, & L. Weixin (Eds.), *Proc. of the 1st Int. Conf. on Evolvable Systems* (pp. 406-422), Berlin, Germany: Springer-Verlag.

Izquierdo-Torres, E. (2004). The Role of Nearly Neutral Mutations in the Evolution of Dynamical Neural Networks. In J. Pollack, M. Bedau, P. Husbands, T. Ikegami, & R. Watson (Eds.), *Proc. of the 9th Int. Conf. on the Simulation and Synthesis of Living Systems* (pp. 322-327), Cambridge MA: MIT Press

Kimura, M. (1983). *The Neutral Theory of Molecular Evolution*. Cambridge, UK: Cambridge Uni. Press

Montana, D. J., & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. *Proc. of the 11th Int. Joint Conf. on Artificial Intelligence* (pp. 762-767), San Mateo, CA: Morgan Kaufmann.

Ortiz-Boyer, D., Hervás-Martínez, C., & García-Pedrajas, N. (2005). CIXL2: A crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research*, *24*, 1-48.

Prechelt, L. (1994). *Proben1 – A Set of Neural Network Benchmark Problems and Benchmarking Rules*. Karlsruhe, Germany: Universtät Karlsruhe (Tech. Report 21).

van Nimwegen, E., & Crutchfield, J. P. (2000). Metastable Evolutionary Dynamics: Crossing Fitness Barriers or Escaping via Neutral Paths? *Bulletin of Mathemetical Biology*, *65*(5), 799-848.

Radcliffe, N. J. (1990). *Genetic neural networks on MIMD computers*. Unpublished D.Phil. thesis. University of Edinburgh, Edinburg, Scotland.

Radcliffe, N. J. (1993). Genetic set recombination and its application to neural network topology optimisation. *Neural Computing and Applications*, *1*(1), 67-90.

Schaffer, J. D., & Morishima, A. (1987). An adaptive crossover distribution mechanism for genetic algorithms. In J. J. Grefenstette (Ed.), *Proc. of the 2nd Int. Conf. on Genetic Algorithms and their application* (pp. 36-40), Cambridge, MA: Lawrence Erlbaum Associates.

Schaffer, J. D., Whitley, D. L., & Eshelman, L. J. (1992). Combinations of genetic algorithms and neural networks: a survey of the state of the art. In D. L. Whitley, & J. D. Schaffer (Eds.), *Proc. Int. Workshop on Combinations of Genetic Algorithms and Neural Networks* (pp. 1-37). Los Alamitos, CA: IEEE Computer Society.

Shipman, R., Shackleton, M., & Harvey, I. (2000). The use of neutral genotype-phenotype mappings for improved evolutionary search. *BT Technology Journal*, *18*(4), 103-111.

Smith, T., Husbands, P., Layzell, P., & O'Shea, M. (2002). Fitness Landscapes and Evolvability. *Evolutionary Computation*, *10*(1), 1-34.

Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*(2), 99-127.

Thierens, D. (1996). Non-Redundant Genetic Coding of Neural Networks. *Proc. of the 1996 IEEE Int. Conf. on Evolutionary Computation* (pp. 571-575), Cambridge, MA: The MIT Press.

Thierens, D., Suykens, J., Vandewalle, J., & Moor, B. D. (1993). Genetic weight optimization of a feedforward neural network controller. In R. F. Albrechts, C. R. Reeves, & N. C. Steel (Eds.), *Artificial Neural Networks and Genetic Algorithms* (pp. 658-663), New York, NY: Springer-Verlag.

Watson, R. A., & Pollack, J. B. (2000). Recombination Without Respect: Schema Combination and Disruption in Genetic Algorithm Crossover. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, & H.-G. Beyer (Eds.), *Proc. of the 2000 Genetic and Evolutionary Computation Conference* (pp. 112-119), San Mateo, CA: Morgan Kaufmann.

Whitley, D. L. (1995). Genetic Algorithms and Neural Networks. In G. Winter, J. Periaux, M. Galan, & P. Cuesta (Eds.), *Genetic Algorithms in Engineering and Computer Science* (pp. 203-126). Chichester, UK: John Wiley & Sons, Ltd.

Whitley, D. L., Starkweather, T., & Bogart, C. (1990). Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, *14*(3), 347-361.

Wolberg, W. H., & Mangasarian, O. L. (1990). Multisurface method of pattern separation for medical diagnosis to breast cytology. *Proc. of the Nat. Academy of Sciences of the USA*, *87*(23), 9193-9196.

Yao, X. (1999). Evolving Artificial Neural Networks. *Proc. IEEE*, *87*(9), 1423-1447.

Yao, X., & Liu, Y. (1997). A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Trans. on Neural Networks*, *8*(3), 694-713.