Multi-Task Learning and Transfer: The Effect of Algorithm Representation

Peter C. R. Lane PETER.LANE@BCS.ORG.UK
School of Computer Science, University of Hertfordshire, College Lane, HATFIELD AL10 9AB, Hertfordshire, England

Fernand Gobet FERNAND.GOBET@BRUNEL.AC.UK
School of Social Science and Law, University of Brunel, UXBRIDGE UB8 3PH, Middlesex, England

Abstract

Exploring multiple classes of learning algorithms for those algorithms which perform best in multiple tasks is a complex problem of multiple-criteria optimisation. We use a genetic algorithm to locate sets of models which are not outperformed on all of the tasks. The genetic algorithm develops a population of multiple types of learning algorithms, with competition between individuals of different types. We find that inherent differences in the convergence time and performance levels of the different algorithms leads to misleading population effects. We explore the role that the algorithm representation and initial population has on task performance. Our findings suggest that separating the representation of different algorithms is beneficial in enhancing performance. Also, initial seeding is required to avoid premature convergence to non-optimal classes of algorithms.

1. Introduction

Meta-learning attempts to understand the interaction between how learning occurs and the contexts in which the learning is applied. We explore a concrete application of meta-learning: the automatic development of a cognitive model taken from multiple competing theories, applicable to multiple experimental tasks. We use a non-dominated sorting genetic algorithm (NDSGA) (Srinivas & Das, 1994) to construct a set of models which are not outperformed on the multiple tasks. Our application is made more complex by the presence of multiple kinds of theories. The main focus of this paper is to explore the effect of the underlying algorithm representation on the composition of the best performing models, and their performance.

Our application attempts to develop a cognitive model applicable to multiple experimental tasks. A cognitive model is a specific example of a class of models (known as a *theory*), and typically is optimised to fit the performance of human participants in an experimental setting. For example, categorisation is a problem requiring objects to be allocated to categories. Humans are asked to categorise objects after seeing some training examples. The performance of the humans becomes the *target* behaviour, which we attempt to model. Theories about categorisation use different learning algorithms to capture the learning process, examples being discrimination-networks and connectionist networks. The cognitive scientist is faced with the task of locating those learning models which perform best on all of the multiple datasets. Our application of meta-learning with NDSGA attempts to automate this task, by learning sets of optimal models.

Optimising the performance of a learning algorithm to a single task is a relatively straight-forward problem in optimisation. Genetic algorithms (GAs) (Holland, 1975) may be used to train a population of examples of the learning algorithm until a high performing example is located. The main idea used by a GA is to sort the evolving population into order of fitness, and then select the best performing members of the population at each turn. Learning multiple tasks presents a different problem. A single algorithm may do well on one task, but less well on a second, whereas a different algorithm may do well on the second task, but less well on the first. The problem is how to directly compare

Appearing in *Proceedings of the ICML-2005 Workshop on Meta-learning*, Bonn, Germany, 2005. Copyright 2005 by the authors.

the performance of algorithms on multiple tasks in a way which will fit onto a linear scale.

Formally, attempting to locate those algorithms which perform well on multiple tasks is known as multi-criteria optimisation. The particular algorithms of interest are those which are not *dominated* by any other. The set of non-dominated algorithms is the *pareto-optimal set*. GAs may be used to locate such pareto-optimal sets, because they manipulate a collection of algorithms simultaneously. Schaffer (1984) was the first to propose the use of GAs in this context. In this paper, we adopt the approach of Srinivas and Das (1994), which uses non-dominated sorting within a genetic algorithm to locate a pareto-optimal set. The idea of NDSGA is to give those algorithms which are non-dominated a high fitness, irrespective of their performance on specific tasks.

The NDSGA manages the interaction between multiple constraints in order to learn an optimal set of algorithms. However, one further challenge is the manner in which a GA combines and uses individuals from its evolving population to create new individuals. In particular, a GA will use cross-over to combine the representations of two separate algorithms. Such a combination is problematic when the algorithms may be of different *types*.

In this paper, we explore the effect that representation has on the constitution and performance of pareto-optimal sets evolved for a specific application of NDSGA. The application is to find those examples drawn from four classes of learning algorithm which best fit distinct and multiple experimental data. In other words, there is a large space of multiple classes of learning algorithms (the cognitive models), and the performance of each algorithm is judged by its ability to match experimental data of multiple kinds.

2. Developing Cognitive Models

An important aspect of cognitive science is the development of computational models which can simulate the behaviour of human experimental participants. Classes of models are collected together into theories, where a set of constraints or typical representations are used to define a collection of similar models. Experimental evidence is obtained on different tasks. As an example, we consider the problem of *categorisation*, which has been studied in intense detail by psychologists and modellers. Initial experiments by Medin and Smith (1981) led to a large number of followup studies. Smith and Minda (2000) describe a collection of thirty experimental results, based on a similar structure.

Table 1. The 5-4 structure used in categorisation experiments (after (Medin & Smith, 1981; Smith & Minda, 2000))

	Attribute (A)			
Example	A0	A1	A2	A3
A examples				
E1	1	1	1	0
E2	1	0	1	0
E3	1	0	1	1
E4	1	1	0	1
E5	0	1	1	1
B examples				
E6	1	1	0	0
E7	0	1	1	0
E8	0	0	0	1
E9	0	0	0	0
Transfer items				
E10	1	0	0	1
E11	1	0	0	0
E12	1	1	1	1
E13	0	0	1	0
E14	0	1	0	1
E15	0	0	1	1
E16	0	1	0	0

The basic experimental structure is known as the 5-4 structure, and is illustrated in Table 1. There are four binary attributes. The task of the learner is, given the examples of categories A and B, to categorise the entire set of examples. If this were a machine learning application, the task would be relatively straightforward: find a learning algorithm whose generalisation error was as small as possible. However, for a psychological experiment, the performance of the learning algorithm is assessed against that of human participants performing the same experiment.

2.1. Tasks

Table 2 provides three kinds of target data for the cognitive model. The first column shows the observed percentage correct for each stimulus in a single experiment. The second column gives the average percentage correct across thirty experiments. The final column gives the time required to provide an answer (time for the transfer items was not measured).

The tasks which a cognitive model must achieve are then determined in a two-fold manner. First, the form of assessment being made is chosen: it may be the (simulated) time required to give an answer, or the probability of giving a correct answer. Second, the data against which the model's performance will be measured is provided. The model's performance will then be assessed: here, we consider the average abso-

Table 2. Target behaviours in 5-4 structure: Probability of responding A, mean number of errors in training, and time to make a classification. (Classification times were not collected for the transfer items.)

	$P(R_A\|E_i)$		
Example	1st	Avg	Time (s)
E1	0.97	0.83	1.11
E2	0.97	0.82	1.34
E3	0.92	0.89	1.08
E4	0.81	0.79	1.27
E5	0.72	0.74	1.07
E6	0.67	0.30	1.30
E7	0.82	0.28	1.08
E8	0.97	0.15	1.13
E9	0.95	0.11	1.19
E10	0.72	0.62	
E11	0.98	0.40	
E12	0.27	0.88	
E13	0.39	0.34	
E14	0.44	0.40	
E15	0.77	0.55	
E16	0.91	0.17	

lute deviation (AAD), and sum-squared error (SSE).

2.2. Cognitive models

We consider four types of cognitive models: two mathematical models, which are known as *context* and *prototype* models; a class of discrimination-network model, known as *CHREST*; and a simple *connectionist* model. Each type of model learns to classify instances of the 5-4 task. The details of the models are not important for this paper, and are not included for reasons of brevity. Some brief details are provided below, and the references provide fuller descriptions.

The two mathematical models are defined in Smith and Minda (2000). They each adopt a weighting between the four attributes, and measure the similarity of an instance to each of the examples in category A or B. The context model measures similarity against all the training examples, whereas the prototype model measures similarity against a single prototype example.

The discrimination-network model, CHREST, is defined in Gobet *et al.* (2001). CHREST is the most complex model considered here, and makes the most serious attempt to model the processes underlying the learning and processing of examples. As a process model, it provides more detailed options for capturing the timing of categorisation, in terms of the number of tests made and the depth of the trained memory.

The connectionist model (e.g. see McLeod et al., 1998) is a simple Hebbian network, using four input units and a single output unit. Time to respond to an input is provided as a single measure of time taken.

3. Evolving a Model

We may treat the four classes of models as defining a space, $\mathcal{M}$, of possible models. Our aim is to find one or more $m \in \mathcal{M}$ satisfying the multiple experimental constraints. We refer to each constraint as a function, mapping a given model to a real number:

$$f_i : \mathcal{M} \to \mathcal{R}$$

Without loss of generality, we assume that we want to minimise $f_i(m) \geq 0$

We say that model m_1 dominates model m_2 if:

$$\forall i \bullet f_i(m_1) \leq f_i(m_2) \ \wedge \ \exists j \bullet f_j(m_1) < f_j(m_2)$$

In other words, m_1 does at least as well as m_2 everywhere, but there is at least one task in which m_1 does better. The set of non-dominated solutions to a multi-criteria optimisation problem is known as the pareto-optimal set. The non-dominated solutions will be those learning algorithms, selected from multiple classes, which are not outperformed on all of the multiple tasks.

We use a non-dominated sorting genetic algorithm (NDSGA) to locate a pareto-optimal set. NDSGA was first proposed by Goldberg (1989), and applied by Srinivas and Das (1994) to learn solutions to mathematical functions. We employ a form of NDSGA, which works as follows:

1. Create an initial population.
2. Form four sets:

   set 1 the non-dominated members of the entire population;

   set 2 the non-dominated members of the remaining population, *i.e.*, the entire population without set 1;

   set 3 the non-dominated members of the remaining population, without set 1 and set 2; and

   set 4 the remaining population, without sets 1, 2, and 3.
3. Create a new population consisting of the members of set 1, and the results of performing crossover on individuals selected from the four sets. The probability of selecting a member of set 3 is

Table 3. Model parameters, and initial maximum random value.

Name		Initial Maximum
Context Model		
Weight 0	C_0	1.0
Weight 1	C_1	1.0
Weight 2	C_2	1.0
Weight 3	C_3	1.0
Sensitivity	C_s	1.0
Guessing	C_g	1.0
Response Time	C_t	100.0
Prototype Model		
Weight 0	P_0	1.0
Weight 1	P_1	1.0
Weight 2	P_2	1.0
Weight 3	P_3	1.0
Sensitivity	P_s	1.0
Guessing	P_g	1.0
Response Time	P_t	100.0
Connectionist Model		
Output threshold	Co_θ	1.0
Learning rate	Co_η	1.0
Attention	Co_a	1.0
Respone Time	Co_t	100.0
CHREST Model		
Attention	CH_a	1.0
Reaction Time	CH_r	100.0
Sorting Time	CH_s	100.0

double that of selecting from set 4; from set 2 double that of selecting from set 3, and from set 1 double that of selecting from set 2.[1]

4. Mutation is performed on the whole population. The probability of changing the model type is *mmt*. If the model type is left alone, one parameter is selected at random, and modified slightly. Table 3 lists the model parameters: a parameter whose initial maximum value is 100 may vary by ± 10, else it may vary by ± 0.1.

5. The process begins again at step 2, until the maximum number of cycles has been reached.

4. Representing the Models

The genetic algorithm manipulates a population of candidate models, where each individual model is represented as a list of values indicating the type of model and its parameter values. As Table 3 illustrates, each model type is defined by a varying number of parameters, each with a distinct role in the model.

In order to perform the generic operations of crossover and mutation, it is important for the genetic algorithm that all the models use the same basic form of representation as a list of items within a genome. What varies between the models is the interpretation placed on each item within its representation. Considering the different models in this application, there are three basic ways of combining the representation:

over-lap Where the first item in the gene represents the model type, and the subsequent items list the parameters for that model type.

separate Where the first item in the gene represents the model type, and distinct parts of the gene are specialised for the parameters of each model type.

mixed Where the first item in the gene represents the model type, and *some* of the gene positions are shared, where parameters perform the same task in two or more different models.

[1]The proportions are arbitary, and aim only to bias the new population to examples of non-dominated algorithms.

Table 4. Usage of gene positions based on representation type. Empty cells indicate that the gene position had no use.

Gene	Over-lap	Separate	Mixed
0	Type	Type	Type
1	$C_0\ P_0\ Co_\theta\ CH_a$	C_0	$C_0\ P_0$
2	$C_1\ P_1\ Co_\eta\ CH_r$	C_1	$C_1\ P_1$
3	$C_2\ P_2\ Co_a\ CH_s$	C_2	$C_2\ P_2$
4	$C_3\ P_3\ Co_t$	C_3	$C_3\ P_3$
5	$C_s\ P_s$	C_s	$C_s\ P_s$
6	$C_g\ P_g$	C_g	$C_g\ P_g$
7	$C_t\ P_t$	C_t	$C_t\ P_t\ Co_t$
8		P_0	Co_θ
9		P_1	Co_η
10		P_2	$Co_a\ CH_a$
11		P_3	CH_r
12		P_s	CH_s
13		P_g	
14		P_t	
15		Co_θ	
16		Co_η	
17		Co_a	
18		Co_t	
19		CH_a	
20		CH_r	
21		CH_s	

Table 4 describes the use made of each gene position in the different representation types. As can be seen, the over-lap condition forces up to four different parameter types to use the same gene position, whereas the separate condition separates out all the parameter types.

The mixed condition attempts to align parameters with a similar role and range to the same position of the gene. For example, both the connectionist and

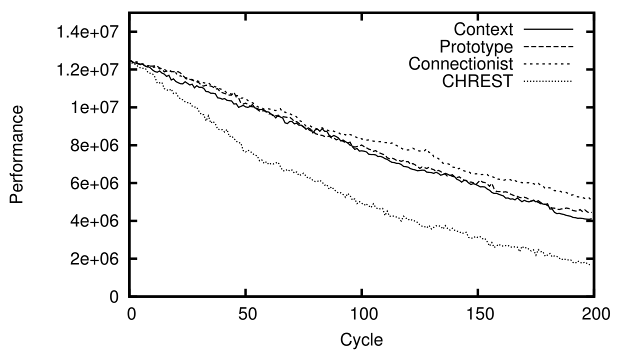

Figure 1. Total performance over training time for all four models

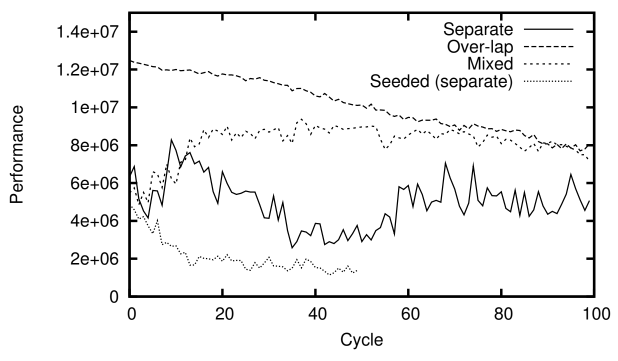

Figure 2. Performance of system against time for different representations.

CHREST models have a probability of attending to a stimulus. Regardless of the separate performance of the models, this attention parameter may be expected to play a similar role in both types. By forcing the parameter to occupy the same position of the gene (10, in this case), any modification of this parameter with respect to either model type will be transferred to the other type in the event of cross-over.

5. Experiments

The main focus of these experiments is to explore the effect of the differing representation types described in Section 4. We also consider how the distribution of model types within the evolving population affects the performance of the final models.

5.1. Baseline performance

The NDSGA algorithm is run with a population of 100 and over 200 cycles four times, each time with a separate class of model type. The probability of mutating the model type, *mmt*, is kept at zero, so that each run is performed with a single type of model. The aim of the experiment is to locate models and performance levels without the effect of cross-model competition.

By summing the values for each of the constraints, we can obtain a measure of how well the system is performing in the aggregate. Figure 1 shows how the summed value varies across the training time of the system. Table 5 shows the best performance on each of the separate tasks.

5.2. Varying representations

The NDSGA algorithm is run with a population of 400 individuals, initially seeded with 100 examples of each

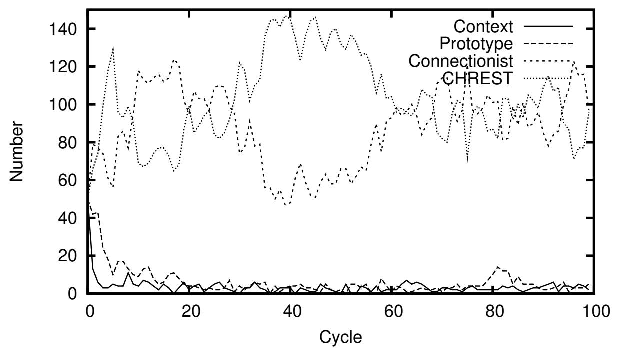

Figure 3. The number of each model type in the entire population, for the separate representation.

of the four classes of cognitive model. The system is trained over 100 cycles, with *mmt* set to 0.05. The system is run three times, each time with a different form of representation, as described in Section 4.

Figure 2, which can be compared with Figure 1, shows the performance of the total set of models as a whole against the training time, with a separate line for each of the representation types. Table 5 shows the best performing model from the final population against each of the separate tasks. Figures 3 and 4 show the numbers of each model class in the complete and non-dominated populations over time for the separate representation only. (One representation only shown for reasons of space.)

5.3. Seeding population

The performance of the mathematical models in the combined populations is relatively poor against the other models. One way to create a more balanced initial pool is to seed the initial population for the NDSGA with individuals taken from the baseline pool.

Table 5. Best performance on each separate task. C – Context Model, P – Prototype Model, Co – Connectionist Model, CH – CHREST Model

Task	Baseline		Over-lap		Separate		Mixed		Seeded	
SSE 1	0.226	C	0.204	Co	0.235	Co	0.222	Co	0.180	C
SSE 2	0.111	C	0.266	Co	0.256	Co	0.256	Co	0.143	C
SSE 3	0.111	C	0.266	Co	0.256	Co	0.256	Co	0.143	C
SSE 4	0.095	C	0.253	Co	0.281	Co	0.249	Co	0.121	C
AAD 1	0.091	P	0.096	Co	0.101	Co	0.103	Co	0.083	C
AAD 2	0.066	C	0.083	Co	0.083	Co	0.083	Co	0.063	C
AAD 3	0.066	C	0.083	Co	0.083	Co	0.083	Co	0.063	C
AAD 4	0.052	C	0.094	Co	0.087	Co	0.082	Co	0.058	C
SSE Avg	0.100	P	0.080	Co	0.161	Co	0.151	Co	0.088	C
AAD Avg	0.066	Co	0.058	Co	0.078	Co	0.075	Co	0.059	C
SSE Time	1,292,901	CH	6,242,291	CH	68,535	CH	79,730	CH	74,966	CH
AAD Time	0.341	CH	0.827	CH	0.069	CH	0.080	CH	0.069	CH

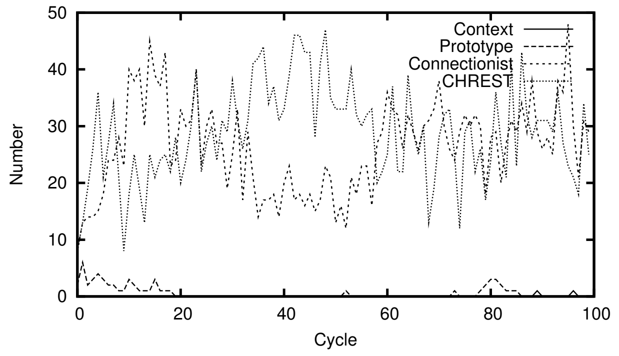

Figure 4. The number of each model type in the non-dominated population, for the separate representation.

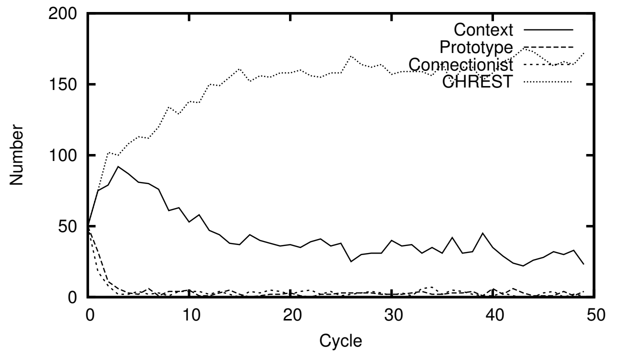

Figure 5. The number of each model type in the entire population, for the separate representation, seeded population.

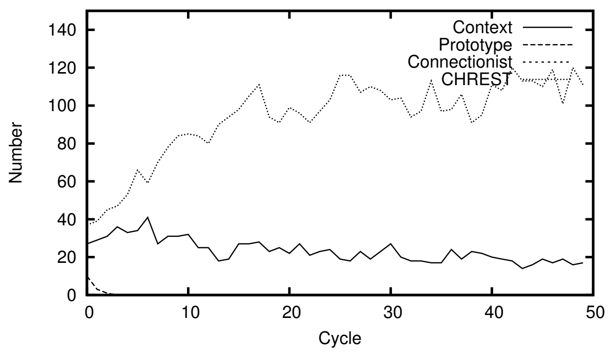

Figure 6. The number of each model type in the non-dominated population, for the separate representation, seeded population.

Essentially, the non-dominated individuals trained from populations with single model types are used to create the initial population for evolving the models in competition with other model types.

Using the separate representation, 50 training cycles were performed with a pool of 200 models, and *mmt* of 0.05. Figure 2 shows the aggregate performance of the seeded population, just for the separate representation, compared with the unseeded populations. Figures 5 and 6 show the proportion of the entire and non-dominated populations of each model type.

5.4. Discussion of experiments

Figure 1 gives the rate of convergence for the four types of models when optimised separately. All four show a similar gradient, or rate of convergence. The better performance of the CHREST type of model is due to its superior performance in the timing tasks, which have a higher 'cost': this kind of scaling problem is typical of the problem arising when attempting to combine multiple criteria into a single criterion.

Figure 2 shows the rate of convergence for the three separate representation types and the seeded population (separate representation only, note that training only proceeded for 50 cycles). It can be seen that the separate representation performs much better than its two colleagues in terms of its aggregate performance. However, the total performance does not approach that of the seeded population.

Figures 3 and 4 illustrate the proportion of each model type present in the entire and non-dominated sets, respectively. The two proportions are similar, and each show a domination of the population by the connectionist and CHREST class of models. These graphs indicate that examples of these models are much easier to locate and preserve in the population than the mathematical models. However, Figures 5 and 6 indicate a different story when the population is seeded with good members from each of the model types. Now, the CHREST and context models dominate the population.

Table 5 explains some of these data. As can be seen, context models mostly outperform the others on the non-time tasks, whereas CHREST has a significant advantage on the time tasks. Obtaining good context models to seed the population enables the context models to outperform connectionist models from the start, and so take over the population. The larger number of CHREST models in the population indicate a greater range of parameter values for the models to perform well, whereas the mathematical models require much tighter bounds on their parameter values.

Looking at the constraints separately, Table 5 indicates that the best overall values are obtained for the seeded population, with separate representation. This suggests, as a conclusion, that it is better to separate the representation of the different algorithms when performing NDSGA, and also that seeding is useful to avoid premature convergence to non-optimal model types. There is little to suggest that combining parameters expected to be similar, as in the mixed representation, has any demonstrable impact on performance.

6. Discussion

6.1. Multi-criteria optimisation

Genetic algorithms are a natural technique for solving multi-criteria optimisation problems (Coello, 2000). However, most of these problems have typically addressed the selection of individuals from a single class. One of the difficulties of this study has been the existence of multiple theories, from which models may be drawn. Few previous studies of such problems exist; notable is that of Kalousis, Gama and Hilario (2004), who consider a similar problem with inductive algorithms. Here, we have addressed a real problem within cognitive science, developing a model of categorisation.

The experiments have shown that the choice of representation is important in assisting the development of effective models. In particular, a naive cross-over function can prevent competing models of different types from becoming viable. However, making different models compete is clearly useful in improving the performance of the models. The clearest example here is that of the timing task, where CHREST's performance was substantially improved when it had to compete with alternative models.

6.2. Cognitive science

One of the important consequences of this work is in formalising the search for optimal computational models. Within cognitive science, few modellers routinely use optimisation algorithms to find optimal models, even with evidence that superior models are found in this way. Some earlier, and related, work has used genetic algorithms to locate optimal training parameters for neural network models (Ritter, 1991) and for more complex ACT-R/PM models (Tor & Ritter, 2004). However, both these previous examples used a single fitness function and a single model type on a single task. We have instead defined a formal framework for specifying and comparing models across multiple domains and theories, and, crucially, demonstrated the validity of meta-learning techniques in locating optimal models. Further discussion of this framework, and its implications for cognitive science, may be found in Gobet and Lane (2005).

6.3. Further work

The main focus of further work will be to extend the number of behavioural tasks and classes of models being considered. These extensions will generate a larger search problem for our genetic algorithm, and we anticipate the need for more sophisticated meta-learning techniques to make the search tractable.

Firstly, we have developed a technique using correlation analysis to locate the optimal values of certain key parameters within the models, and so constrain the evolution of future models to those optimal values (Lane & Gobet, 2005). This technique will be extended to include more complex sets of correlations, which may require stronger machine learning techniques to formulate hypotheses about which algorithms are performing the best, and why. Michalski (2000) has already shown that such stronger machine learning techniques provide good results over weaker evolution methods.

Secondly, we aim to exploit the algorithm's search through multiple tasks and multiple theories to locate relations between specific algorithm properties and task performance. For example, it is well known that short-term memory size is important and relatively constant (Cowan, 2001). Our system should be able to come to such conclusions by aggregating evidence about short-term memory size across all the competing model types. Comparisons may be drawn with the analysis of inductive algorithms against datasets conducted by Kalousis *et al.* (2004), where similarities between algorithms and datasets were analysed using error correlation and relative performance.

7. Conclusion

We have presented a novel application of NDSGA to the task of finding optimal cognitive models, where the models must be chosen from multiple classes of competing theories, and also each model is evaluated against multiple constraints. Exploring the effect of initial population seeding and representation type suggests that competition between model classes is beneficial in locating better models, but that separate representations and an initial population seeded with good models are required for NDSGA to perform well.

Acknowledgements

The authors would like to thank the programme chair and three anonymous reviewers for their comments on an earlier version of this paper.

References

Coello, C. A. C. (2000). An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, *32*.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, *24*, 87–185.

Gobet, F., & Lane, P. C. R. (2005). A distributed framework for semi-automatically developing architectures of brain and mind. *Proceedings of the First International Conference on e-Social Science.*

Gobet, F., Lane, P. C. R., Croker, S. J., Cheng, P. C.-H., Jones, G., Oliver, I., & Pine, J. M. (2001). Chunking mechanisms in human learning. *Trends in Cognitive Science*, *5*, 236–243.

Goldberg, D. E. (1989). *Genetic algorithms in search optimization and machine learning.* Reading, MA: Addison-Wesley.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: The University of Michigan Press.

Kalousis, A., Gama, J., & Hilario, M. (2004). On data and algorithms: Understanding inductive performance. *Machine Learning*, *54*, 275–312.

Lane, P. C. R., & Gobet, F. (2003). Developing reproducible and comprehensible computational models. *Artificial Intelligence*, *144*, 251–63.

Lane, P. C. R., & Gobet, F. (2005). Discovering predictive variables when evolving cognitive models. *Proceedings of the Third International Conference on Advances in Pattern Recognition.*

McLeod, P., Plunkett, K., & Rolls, E. T. (1998). *Introduction to connectionist modelling of cognitive processes.* Oxford, UK: Oxford University Press.

Medin, D. L., & Smith, E. E. (1981). Strategies and classification learning. *Journal of Experimental Psychology: Human Learning and Memory*, *7*, 241–253.

Michalski, R. S. (2000). Learning evolution model: Evolutionary processes guided by machine learning. *Machine Learning*, *38*, 9–40.

Ritter, F. E. (1991). Towards fair comparisons of connectionist algorithms through automatically optimized parameter sets. *Proceedings of the Annual Conference of the Cognitive Science Society* (pp. 877–881). Hillsdale, NJ: Lawrence Erlbaum.

Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms.* Doctoral dissertation, Vanderbilt University, Nashville.

Smith, J. D., & Minda, J. P. (2000). Thirty categorization results in search of a model. *Journal of Experimental Psychology*, *26*, 3–27.

Srinivas, N., & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, *2*, 221–248.

Tor, K., & Ritter, F. E. (2004). Using a genetic algorithm to optimize the fit of cognitive models. *Proceedings of the Sixth International Conference on Cognitive Modeling* (pp. 308–313). Mahwah, NJ: Lawrence Erlbaum.