Exploring the Diversity of Evolved Cognitive Models with Cluster Analysis

Peter C.R. Lane[1][0000−0002−8554−2217], Noman Javed[2][0000−0001−7770−6616], Laura K. Bartlett[2][0000−0001−5202−4504], Dmitry Bennett[2][0000−0003−1057−6508], and Fernand Gobet[2][0000−0002−9317−6886]

[1] Department of Computer Science, University of Hertfordshire, Hatfield, UK
p.c.lane@herts.ac.uk (*)
[2] Centre for Philosophy of Natural and Social Science, London School of Economics, Houghton Street, London WC2A 2AE, UK n.javed3@lse.ac.uk, l.bartlett@lse.ac.uk, d.bennett5@lse.ac.uk, f.gobet@lse.ac.uk

Abstract. Cognitive scientists often represent theories of cognitive behaviour in the form of computer programs which simulate and model the performance of humans in experimental settings. Earlier work has demonstrated that evolutionary techniques, specifically genetic programming (GP), can be used to generate a pool of candidate models in the form of executable computer programs. However, previous work has not considered the impact of changes to hyper-parameter values, such as those controlling the behaviour and timing of operators or those controlling the operation of the GP process. In this paper, we develop and use a cluster analysis technique based around the Silhouette Index to investigate the impact of hyper-parameter changes on the composition of evolved populations of programs. Our metrics support visualisations, and enable a user to assess both qualitatively and quantitatively the diversity of candidates from different populations. In this way, a cognitive scientist can analyse the output of the evolutionary system in order to uncover or inspire potentially novel theories of human behaviour.

Keywords: cluster analysis, cognitive science, evolutionary computing, genetic programming, silhouette index, visualisation

1 Introduction

Genetic programming (GP) [7] is a popular technique used to synthesise programs to solve a given problem. In many applications, what matters most is that the given problem is *solved*, in the best possible manner. However, in this paper we will focus on a separate issue: the *diversity* of the solution space. Cognitive scientists often represent their theories of human behaviour in the form of computer programs, from small [11] to large [1, 4]. Computer programs provide many advantages over verbal theories for representing cognitive theories, not least being that their execution provides empirical predictions of behaviour which can be compared with that of humans [6, 10]. However, developing and testing such

computer programs can be a challenging task. Apart from the choice of competing theoretical frameworks, such as whether to use symbolic or connectionist or Bayesian theories, there is also the diverse behaviour shown by humans on identical stimuli; and then there is the process of writing the program itself.

This paper forms part of a larger project to develop AI-based tools to assist a (cognitive) scientist when developing computational models. Here, we focus on two objectives. The first objective is to take a set of theoretical assumptions and experimental definitions from the scientist and demonstrate that at least one solution is possible. This is a straight-forward exercise in program synthesis, using a program to construct a program to meet a problem specification. The second objective is to provide a tool with which the scientist can explore the diversity of the solution space. This diversity can arise from two factors. First, as mentioned above, humans often exhibit diverse behaviour on identical stimuli: where one person given an image will focus on a single aspect of the image, a second will allow their gaze to wander across the image and even around the room. Second, some people have differences in their cognitive abilities, such as a loss of short-term memory capacity with age, or differences in reaction times. Diversity of the first kind should be found automatically by our program-construction process; diversity of the second kind will be prompted by the cognitive scientist modifying parameter settings which affect the kinds of constructed program.

Previous work [2, 8, 9] has already demonstrated the first objective. Given a theoretical framework, such as the definition of a simple symbolic architecture, and an experimental setting, such as that of the Delayed Match to Sample (DMTS) task [3], candidate computational models can be created using GP with excellent fits to the empirical data. In this paper, we examine the diversity of the solution space. When GP runs, it works in a stochastic manner, and often its population converges on a selection of very similar kinds of model. By rerunning the GP process with the same parameters, often quite different kinds of model can be generated. In addition, by changing some of the hyper-parameters, whether these are part of the definition of the computational models or the GP search process, different models again can be generated. The question then is, are these different sets of models qualitatively different solutions from each other, or merely variations on a theme? Apart from manual examination, are there metrics or visualisations which can help a cognitive scientist explore the diversity of candidate models generated by GP?

Our answer to these questions rests on two principles. First, we define a measure of similarity between two programs [8]. This measure allows us to use cluster analysis tools, such as the Silhouette Index [12], to provide a metric comparing two groups of models: the metric quantifies the similarity of the two groups. Second, this measure allows us to generate a visualisation of the space of found solutions, with distance approximating model similarity. Such a visualisation is ideal for navigating and exploring different models, seeking insights into theoretical mechanisms.

In the remainder of this paper we introduce our model synthesis system, known as GEMS (Genetically Evolving Models in Science). This involves the def-

Fig. 1. Illustration of DMTS task. (Photos by Danny de Bruyne and Ronaldo Taveira, freeimages.com)

inition of the model architectures, the experimental task to fit, and the GP search algorithm. We then introduce the similarity measure used to compare models and describe how we will use the Silhouette Index to compare groups of models. We present some simulation experiment results in which we consider different hyper-parameter settings and explore the diversity of the generated models. In particular, we consider theoretical implications of the generated models.

2 Overview of GEMS

Our GEMS system for automatically developing cognitive models is an example of *program synthesis*. Such systems can be conveniently divided into three parts [5]: the task definition (*user intent*), to express what makes a good program; a *search space* of candidate programs; and a *search technique*, to explore the given search space for good programs. Here, the developed programs form the control structure for the cognitive models.

2.1 Task definition: DMTS

As an example task, we take the Delayed Match to Sample (DMTS) task [3]. This task is a typical neuro-scientific experiment used in studies on the accuracy and reaction time of short-term memory. The experiment is illustrated in Fig. 1. The participant is initially shown an image for 1 second, known as the 'target', before it is removed. After a delay of 0.5 seconds, two 'simulus' images are presented for 2 seconds, one on the left and the other on the right. Once the stimuli appear, the participant must select either 'left' or 'right' as their response, indicating which of the simuli is the same as the target; this response must be made as quickly as possible.

The challenge of this task lies not only in obtaining an accurate response, but in providing a model of the human behaviour. To do this, we must fit the performance of human participants: an accuracy of 95.7% and an average response

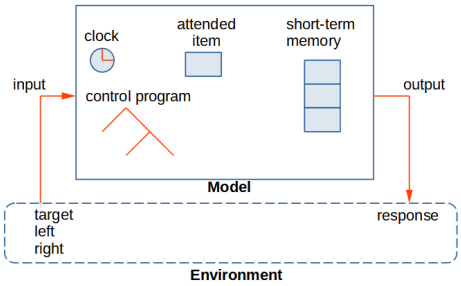

Fig. 2. Overall structure of cognitive model.

time of 767ms [3]. So firstly, the best model is not one that gets all responses correct, but one that has an accuracy of 95.7% – corresponding to about two errors out of the entire set of tests. Secondly, the model must coordinate its perception of time-sensitive information (the presence of the target and stimuli images) with making a response within an expected response time of 767ms.

2.2 Search space: Cognitive models

A cognitive model is a kind of virtual machine representing a simulated human. As shown in Fig. 2, the model interacts with an external environment. It has a clock, representing the simulation time, and two internal memory structures: a short-term memory (STM), which is a fixed-size queue of items, and an attended item slot, used to hold temporary information, including the results of comparison operations. The model has an internal control program which coordinates and determines its actions.

The control program is composed from various operators, which include: observing the target slot (input-target), observing the left or right stimulus (input-left or input-right), preparing to output a response (respond-left or respond-right), placing items into STM (put-stm) or comparing items in STM (e.g. compare-1-2 compares the top two slots in STM, setting the attended item slot to 1 if they are equal or 0 if not), and some syntactic elements such as an if-statement: the if-statement will move to its true or false block depending on if the current value of the attended-item slot is 1 or 0, respectively.

Apart from their effect on the memory structures within the model, each operator also requires a certain amount of simulated time to act, which is added to the clock time: e.g. an input operation requires 100ms, a STM operation 50ms, or a cognitive operation 70ms. (These timings are investigated in the simulation experiments, presented later.)

2.3 Search technique: Genetic programming

The basic structure of our models is fixed, as in Fig. 2. We obtain different models by providing different control programs. This could, of course, we done by

hand. However, the aim of GEMS is to automate this task, particularly to find multiple possible models which fit the empirical data. In our project, we use Genetic Programming (GP) [7] to generate candidate models using an evolutionary search algorithm. GP works by first creating a population of many models. It then ranks the models based on how well they complete the set task (using a "fitness function") and chooses the best models to generate a new population of models: the new models are generated by changing small parts of existing models (mutation), or by combining parts of two existing models (crossover). The GP process then repeats for a number of cycles.

An important feature of the fitness function used in this paper is that it is composed of three elements: one for accuracy, one for reaction-time and one for program size. Formally, the three components of the fitness function are:

1. $f_a = |\text{accuracy} - 0.957|/0.957$: this is the difference of the model's and target accuracy, scaled to the range $[0, 1]$.
2. $f_t = \text{half-sigmoid}(|\text{response-time} - 767|/\text{RT})$: this is the difference of the model's and target response time, with a variable scale factor RT.
3. $f_s = \text{half-sigmoid}(|\text{program-size} - 10|/\text{PS})$: this is the difference of the model's and an arbitrary target program size of 10, with a variable scale factor PS.

where $\text{half-sigmoid}(x) = 2 \times (1/(1 + e^{-x}) - 0.5)$ is the usual sigmoid function which we rescale from [0.5,1] to [0,1], because all our values of x are positive. The variable scale factors in f_t and f_s control the steepness of the sigmoid slope.

The *overall fitness* is computed as a combination of these three, with multipliers $a + b + c = 1$ ensuring that the overall fitness is in the range $[0, 1]$:

$$f = a \times f_a + b \times f_t + c \times f_s \tag{1}$$

See Lane et al. [8] for more details on our use of GP, and particularly the post-processing that is used on the models to remove duplicates.

3 Quantifying Model Diversity

The Silhouette Index [12] is a measure of the consistency of data clusters. The measure takes a set of data points and their arrangement into clusters, along with a pair-wise distance function for points, and returns a number between -1 to 1 representing the consistency of the clusters: a value of 1 indicates that the clustering is good, with members of each cluster far from members of other clusters.

The Silhouette Index is computed in this paper as follows:

1. $a(i)$ is the average distance of instance i to all other points within its cluster
2. $b(i)$ is the smallest over the other clusters of the average distance of instance i to all other points within the other cluster
3. $s(i)$ is computed as

$$\frac{b(i) - a(i)}{\max(b(i), a(i))} \tag{2}$$

4. The Silhouette Index, $\overline{s(i)}$ is the average $s(i)$ for all instances

In this paper, the index is only used to compare two clusters, so $a(i)$ will represent the average distance of an instance to points within its own cluster and $b(i)$ will represent the average distance of an instance to points within the other cluster.

As can be seen from step 3, a value of -1 occurs when $b(i)$ is zero, meaning the average distance to instances in the other cluster is 0: this means the instance is located within the other cluster. Conversely, a value of 1 occurs when $a(i)$ is zero, meaning the average distance to instances in its own cluster is 0: this means the instance is located at the centre of its own cluster. Positive values correspond to clusters which are well separated, meaning that the instances in the two clusters are different and more diverse.

For a distance measure between two models, we compute the similarity of their control programs. Each program is separated into a set of node+child-labels segments. For example, the following program is converted into eight segments of two parts and six individual node names:

```
(if (compare-1-2)
    (prog2 (input-right) (input-left))
    (input-target))

parts: (if compare-1-2 prog2 input-target)
       (prog2 input-right input-left)
names: if compare-1-2 prog2 input-target
       input-right input-left
```

The pair-wise similarity (Jaccard Index) divides the number of common segments in the two programs (the set intersection) by the total number of segments (the set union): distance is then computed as $1 -$ similarity.

4 Simulation Experiments

The aim of these simulation experiments is to explore the diversity of computational models created by GP under different hyper-parameter settings. The diversity will be measured qualitatively using visualisations and quantitatively using a Silhouette Index score, as outlined in the previous section.

Each hyper-parameter setting is used to generate a population of models. The GP system is run using a population size of 5000 for 250 generations. The "good models" are defined as those with an overall fitness less than or equal to 0.1. The first setting is the base-line setting, using parameter values from previous publications and, in particular, the operator timings are based on best estimates from the psychological literature, as described in Section 2.2: for this first setting, good models are obtained from five runs of the GP system, to obtain a diverse collection. For all other settings, a single run of the GP system is made.

The hyper-parameter settings are defined as follows (bold names will be used to refer to each setting in the remainder of this section, all settings are changed with relation to the base setting):

base uses the full set of operators and these settings:
- time settings: input=100, output=140, cognitive=70, stm=50, syntax=0
- fitness function proportions: a=0.7, b=0.2, c=0.1 (these approximately make reation time twice as important as program size, and accuracy twice as important as them combined)
- Reaction Time fitness parameter: time-rt=767

timings-1 Investigate changes to time settings with low input/output times but high cognitive and STM times: input=10, output=20, cognitive=140, stm=200, syntax=50

timings-2 Investigate changes to time settings with low input and cognitive times but high output and STM time: input=10, output=200, cognitive=10, stm=200, syntax=100

time-rt Investigate impact of changing the parameter for the reaction-time squashing function: time-rt=100

no-cmp-1-2 Investigate impact of a change to cognitive capabilities: removes compare-1-2 operation so models cannot compare the top items in STM.

no-stm Investigate impact of a change to cognitive capabilities: removes the option to add items to STM.

fitness-abc Investigate the hyper-parameters controlling the fitness function: change proportion settings: a=0.2, b=0.3, c=0.5.

4.1 Results

Table 1 reports the performance of the good models found from each setting. After the GP run finishes, the "good models" with a fitness ≤ 0.1 are extracted. The left-side of the table reports some quantitative statistics for the complete set of good models. The right-side of the table reports statistics restricted to the best 50 models.

As can be seen, those settings where models could be found supported high-quality models. For example, setting no-cmp-1-2, where one of the cognitive

	Good Models					Top 50 Models				
Name	Count	Min	Max	Mean	StdDev	Count	Min	Max	Mean	StdDev
base	1603	0.040	0.100	0.062	0.013	50	0.040	0.042	0.041	0.001
timings-1	124	0.037	0.092	0.056	0.015	50	0.037	0.043	0.039	0.002
timings-2	1740	0.047	0.100	0.052	0.010	50	0.047	0.047	0.047	0.000
time-rt	829	0.052	0.100	0.069	0.015	50	0.052	0.052	0.052	0.000
no-cmp-1-2	848	0.043	0.100	0.053	0.007	50	0.043	0.043	0.043	0.000
no-stm	0					0				
fitness-abc	0					0				

Table 1. Performance statistics for all 'good' (fitness ≤ 0.1) and top 50 models.

	base	timings-1	timings-2	time-rt	no-cmp-1-2
mean SI	0.151	0.291	0.150	0.206	0.139

Table 2. Mean SI for the top 50 models from each setting against all others.

	base	timings-1	timings-2	time-rt	no-cmp-1-2
base	1.000	0.504	0.328	0.406	0.352
timings-1	0.504	1.000	0.563	0.671	0.574
timings-2	0.328	0.563	1.000	0.467	0.382
time-rt	0.406	0.671	0.467	1.000	0.520
no-cmp-1-2	0.352	0.574	0.382	0.520	1.000

Table 3. Mean SI for the top 50 models from each pair of settings.

operators was removed, still obtained models with a minimum overall fitness of 0.043 compared to the base setting's minimum overall fitness of 0.040.

There were two settings where no good models were found: these are no-stm and fitness-abc. It is not surprising that the no-stm setting failed to find any good models: the task cannot be completed without comparing the current stimuli with the initial stimulus, and this comparison can only happen if the initial stimulus is stored in some kind of memory. However, it was unexpected that changes to the relative weighting of terms in the fitness function failed to generate good models, indicating a sensitivity here to these particular hyper-parameters.

Overall, these results support the first objective of our system, to develop good computational models. To consider the second objective, exploring the diversity of the solution space, we have used the Silhouette Index (SI) as a measure of similarity between the models developed for each setting.

Table 2 shows the SI value of the top 50 models from each setting when compared to models from all the other settings: settings which did not generate any models have been excluded.

All five settings show a positive SI in comparison to all other settings: a positive SI means the clusters are generally well separated, with a higher SI corresponding to greater dissimilarity. The two most dissimilar groups are timings-1 and time-rt.

Table 3 offers a similar analysis, but this time compares the top 50 models from each setting against those from each other setting individually: settings which did not generate any models have been excluded.

The pairwise comparisons again reinforce the impression from the previous table that each setting generates a distinct group of models. Every pair is dissimilar to every other pair, with the highest dissimilarities being between models in the timings-1 group and those in the other groups.

Fig. 3 provides a visualisation of the similarities of the top 50 models from all settings which generated models. The two axes represent each of the 250 models,

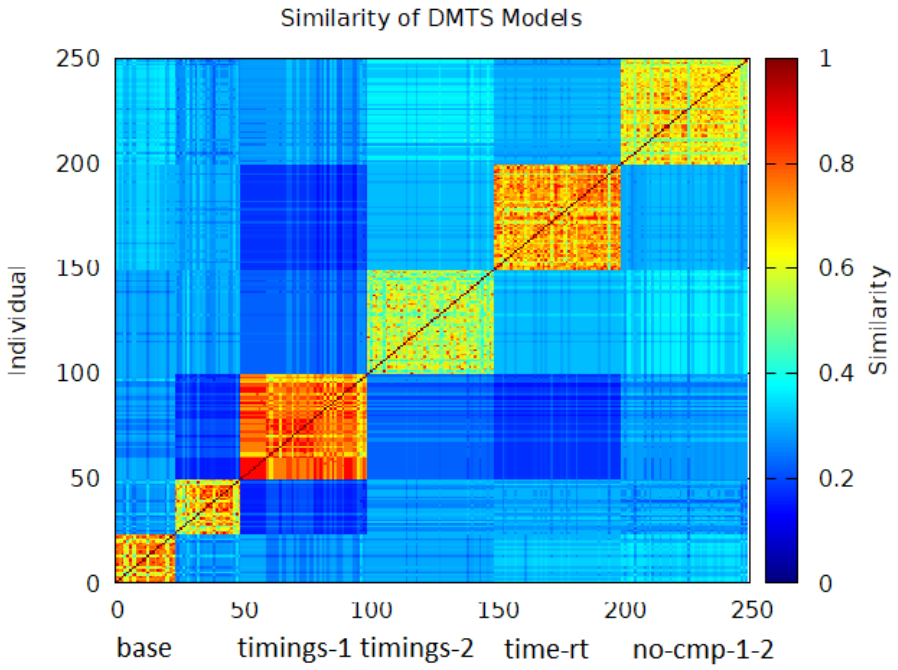

Fig. 3. Similarities of the top 50 models from all settings.

formed from the top 50 models in each of the five different settings. The within setting comparisons show high similarity – a preponderance of yellow and red colour. The between setting comparisons show low similarity – a preponderance of blue colour. As is readily apparent, the models from most of the settings are quite distinct from each other. The first 50 models are drawn from the base setting and there are two distinct groups here: this is because the base setting models are collected from five separate runs of the model-construction process.

A useful kind of visualisation for exploring the diversity of models takes the pair-wise model similarities and converts these into 2D coordinates using multidimensional scaling [13]. These coordinates can then be plotted, as shown in Fig. 4. This chart demonstrates clearly some of the conclusions from our quantitative results. In particular each of the five model groups is located in a distinct area of the chart, with little apparent overlap. Also note the relative isolation of the timings-1 group and, to a lesser extent, the time-rt group: this isolation is indicated in the pairwise Silhouette Index measurements.

4.2 Analysis

The results above demonstrate the diversity of candidate models generated by the GEMS system. Some theoretical insights can be obtained by looking in detail

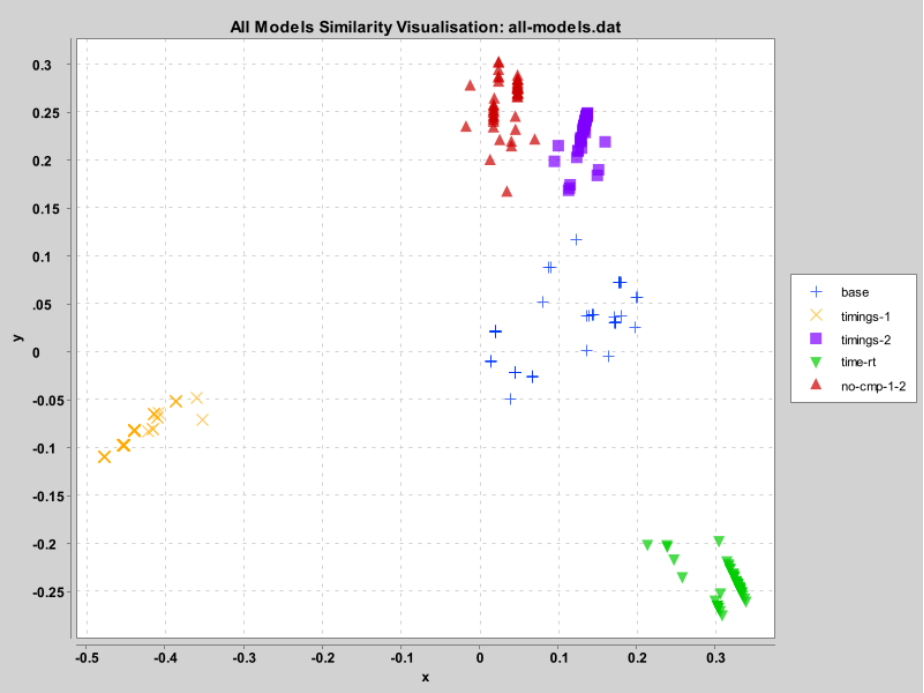

Fig. 4. Models plotted in 2D, where distance is proportional to similarity.

at examples of the generated models. Both the quantitative and the qualitative results suggest that the base, timings-2 and no-cmp-1-2 settings are relatively similar to each other, and the timings-1 and time-rt settings are relatively dissimilar. By picking examples of the dissimilar groups, we can see what kind of model the system has generated in each case.

Fig. 5 shows the best performing model from the base and timings-1 settings, respectively. The two models use different strategies to accomplish the task. The difference can be seen from the conclusion of the second model which has arranged its code into three blocks, in an if-statement. The condition block of the if-statement does all the image scanning and comparison. The result of this condition block then decides whether the model responds left or right, as the last thing it does.

In contrast, the first model sets up a decision to respond "right" in the early part of its program. The scanning of the images is then followed by an if-statement: this if-statement is used to "change the model's mind" – if the contents of STM indicate that "right" is the wrong answer, the model will now respond "left".

We thus have two different strategies: one strategy makes an early decision and then looks for reasons to change its mind, the second strategy waits until it has to output its decision, and only then makes its choice.

```
(PROG4
 (DOTIMES-3
  (PROG2
   (PROG4 (INPUT-TARGET) (WAIT-25) (INPUT-TARGET) (RESPOND-RIGHT))
   (PROG3 (INPUT-TARGET) (INPUT-LEFT) (PUT-STM))))
 (WAIT-200) (COMPARE-1-3)
 (IF (COMPARE-1-3)
  (RESPOND-LEFT)
  (WAIT-25)))

(IF (DOTIMES-5
       (IF (IF (WAIT-25)
            (COMPARE-2-3)
            (PROG4 (INPUT-TARGET) (INPUT-RIGHT) (WAIT-25)
             (INPUT-RIGHT)))
        (PUT-STM)
        (COMPARE-1-3)))
   (RESPOND-RIGHT)
   (RESPOND-LEFT))
```

Fig. 5. Best performing models from the base and timings-1 settings.

This choice of strategy affects the decision-making process. Other models (not shown here) demonstrate different visual-attention strategies, such as whether to scan all image locations all the time, or whether to first scan the target location, and then later move on to the left-right images.

5 Conclusion

We have explored the use of the Silhouette Index cluster analysis metric as a way to examine the diversity of generated models from the GEMS scientific discovery system. In particular, we have examined the diversity of models generated from different runs of the system with different parameters and demonstrated how different strategies have been located: different strategies are evident both in decision making (shown above) and in visual attention. The presented qualitative and quantitative results can be used by a cognitive scientist as a way to uncover or inspire potentially novel theories. The range of theories developed by GEMS and its internal diversity is intractable for a human scientist to develop without computational assistance. AI systems such as GEMS will increasingly become an indispensable aid for all scientists, not only to create individual solutions, but also to explore the full potential of their theories. In future work we will extend our coverage of DMTS-based experiments to cover more examples of the paradigm and better model the visual recognition process through the use of trained neural-networks.

More information about the GEMS project and the software underlying the experiments in this paper can be found at: `https://gems.codeberg.page`

Acknowledgements This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. ERC-ADG-835002).

References

1. Anderson, J.R., Lebière, C. (eds.): The Atomic Components of Thought. Lawrence Erlbaum, Mahwah, NJ (1998). https://doi.org/10.4324/9781315805696
2. Bartlett, L.K., Pirrone, A., Javed, N., Lane, P.C.R., Gobet, F.: Genetic programming for developing simple cognitive models. In: Goldwater, M., Anggoro, F.K., Hayes, B.K., Ong, D.C. (eds.) Proceedings of the 45th Annual Meeting of the Cognitive Science Society. pp. 2833–2839 (2023)
3. Chao, L., Haxby, J., Martin, A.: Attribute-based neural substrates in temporal cortex for perceiving and knowing about objects. Nature Neuroscience **2**, 913–20 (1999). https://doi.org/10.1038/13217
4. Gobet, F., Simon, H.A.: Five seconds or sixty? Presentation time in expert memory. Cognitive Science **24**, 651–82 (2000). https://doi.org/10.1016/S0364-0213(00)00031-8
5. Gulwani, S.: Dimensions in program synthesis. In: Proceedings of the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming. pp. 13–24 (2010). https://doi.org/10.1145/1836089.1836091
6. Jones, G.A., Gobet, F., Freudenthal, D., Watson, S.E., Pine, J.M.: Why computational models are better than verbal theories: the case of nonword repetition. Developmental Science **17**, 298–310 (2014). https://doi.org/10.1111/desc.12111
7. Koza, J.R.: Genetic Programming: On the programming of computers by means of natural selection. MIT Press, Cambridge, MA (1992). https://doi.org/10.1007/BF00175355
8. Lane, P.C.R., Bartlett, L.K., Javed, N., Pirrone, A., Gobet, F.: Evolving understandable cognitive models. In: Stewart, T. (ed.) Proceedings of the 20th International Conference on Cognitive Modelling. pp. 176–182. University Park, PA: Applied Cognitive Science Lab, Penn State (2022)
9. Lane, P.C.R., Sozou, P.D., Gobet, F., Addis, M.: Analysing psychological data by evolving computational models. In: Wilhelm, A., Kestler, H. (eds.) Analysis of Large and Complex Data. Studies in Classification, Data Analysis, and Knowledge Organization. pp. 587–97. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-25226-1_50
10. Lane, P., Gobet, F.: A theory-driven testing methodology for developing scientific software. Journal of Experimental and Theoretical Artificial Intelligence **24**, 421–56 (2012). https://doi.org/10.1080/0952813X.2012.695443
11. Pirrone, A., Lane, P.C.R., Bartlett, L.K., Javed, N., Gobet, F.: Heuristic search of heuristics. In: Bramer, M., Stahl, F. (eds.) Artificial Intelligence XL. vol. 14381, pp. 407–420 (2023). https://doi.org/10.1007/978-3-031-47994-6_36
12. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**, 53–65 (1987). https://doi.org/10.1016/0377-0427(87)90125-7
13. Torgerson, W.S.: Multidimensional scaling. I: Theory and method. Psychometrika **17**, 401–419 (1952). https://doi.org/10.1007/BF02288916