Heuristic search of heuristics

Angelo Pirrone[1,3][0000−0001−5984−7853], Peter C.R. Lane[2][0000−0002−8554−2217], Laura Bartlett[1][0000−0001−5202−4504], Noman Javed[1][0000−0001−7770−6616], and Fernand Gobet[1][0000−0002−9317−6886]

[1] Centre for Philosophy of Natural and Social Science, London School of Economics and Political Science, UK
{a.pirrone,l.bartlett,n.javed3,f.gobet}@lse.ac.uk
[2] Department of Computer Science, University of Hertfordshire, UK
p.c.lane@herts.ac.uk
[3] Department of Psychology, University of Liverpool, UK

Abstract. How can we infer the strategies that human participants adopt to carry out a task? One possibility, which we present and discuss here, is to develop a large number of strategies that participants could have adopted, given a cognitive architecture and a set of possible operations. Subsequently, the (often many) strategies that best explain a dataset of interest are highlighted. To generate and select candidate strategies, we use genetic programming, a heuristic search method inspired by evolutionary principles. Specifically, combinations of cognitive operators are evolved and their performance compared against human participants' performance on a specific task. We apply this methodology to a typical decision-making task, in which human participants were asked to select the brighter of two stimuli. We discover several understandable, psychologically-plausible strategies that offer explanations of participants' performance. The strengths, applications and challenges of this methodology are discussed.

Keywords: Decision Making · Heuristic Search · Program Synthesis · Genetic programming.

1 Introduction

One aim of cognitive scientists is to understand the strategies (heuristics) that human participants adopt in order to solve a problem; whether it is a 'simple' problem, such as deciding to accept or reject a bet, or a more complex problem, such as planning the next move during a chess game.

Understanding which strategies participants use is an inferential problem. No unique procedure exists to infer theories of cognitive capacities, and, in general, this is a problem that can be addressed in several different ways. For instance, based on previous data and on principles such as computability or parsimony, a researcher may hypothesise a specific strategy and design experiments that test qualitative or quantitative predictions of that strategy.

Another approach is to consider a large number of possible strategies and select the best ones on the basis of principles such as goodness of fit and/or simplicity. This process is known as a heuristic search in the space of possible alternatives [19], a principle that AI pioneers Herbert Simon and Allen Newell proposed as an explanation of how humans and computers solve problems and make scientific discoveries.

Here we present and apply a methodology [3, 10, 2] that supports researchers in considering a large range of strategies, built from a set of minimal building blocks of cognitive processes. In this paper, strategies are represented as computer programs, running on a virtual machine embodying a cognitive model of human behaviour. First, a set of strategies is created at random; subsequently, these strategies (i.e., computer programs) are modified using evolutionary techniques, in order to minimise a fitness function. In the current case, fitness is assessed using mean reaction time and mean choice. This yields many strategies that quantitatively explain the data. After post-processing, the number of candidate programs is significantly reduced and insights can be gained regarding the processes underlying the observed behaviour.

Automated theory/program/equation discovery based on evolutionary approaches and other algorithms for combinatorial search has a long history in science [8, 2, 6]; however, applications to cognitive science are rare (but see [21, 16, 12, 20]), especially in the form of symbolic, more easily understandable, cognitive architectures [3, 10]. In the following section, we present an application of this methodology to a decision-making task. We are especially interested in discussing the strengths, challenges and limitations of this methodology with regard to aiding inference in cognitive science.

2 A value-sensitive decision-making task

We consider a typical decision-making task [17, 22, 13]; participants need to decide which of two gray patches presented on a computer screen is brighter, Figure 1. Specifically, during each trial, participants are presented with a fixation cross at the centre of the screen, and, equidistant from the fixation cross, two stimuli, in the form of homogeneous, round, grey patches on a black background. The brightness of each patch is sampled from a normal distribution and varies from frame to frame. That is, the patches have momentary, normally distributed, fluctuations in brightness - to make a choice, participants are assumed to integrate evidence over time to decide which patch is brighter. In other words, to decide which is brighter, participants build up an 'average feel' regarding the brightness of each patch. Participants respond in their own time, meaning that the patches remain on screen until they make a response. To select which alternative is brighter, participants press 'left' or 'right' on a keyboard. Shortly after responding, participants are presented with a new trial.

Crucially, in diagnostic conditions, the two stimuli have identical brightness – these conditions are labelled 'equal alternatives'. Across 'equal alternatives' conditions, the experimenter varies the absolute brightness. That is, physical

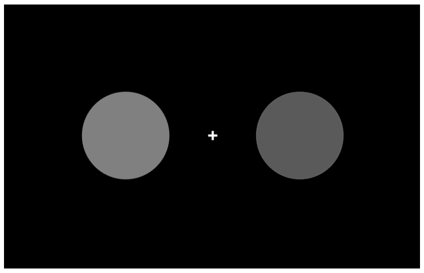

Fig. 1. Stimuli example. Participants need to decide, by pressing one of two buttons on a keyboard, which of the two stimuli patches is brighter. In this example, stimulus patch size is exaggerated for visibility.

and psychological differences between stimuli are kept constant (i.e., zero, since they are identical) while their absolute value is increased or decreased. Note that, when deciding which of the two stimuli is brighter, their absolute brightness is irrelevant and the only relevant feature should be relative brightness. Nonetheless, results show that decisions are made faster as absolute value is increased, even though participants are required to choose the brighter of the two stimuli and the difference between the two stimuli is null [18]. This result is replicated across very different types of choices (e.g., between consumer items, foraging scenarios, etc.) and species, such as non-human primates [17], and highlights a hard-wired property of our decision architecture – value-sensitivity (for details on value-sensitivity, see [18]). Note that 'value' is not to be confused with 'reward' – here 'value' only refers to the intensity or magnitude of the stimuli. Value-sensitivity, in which decisions are made faster and more random (i.e., less accurate) as absolute input value increases, challenges dominant theories of decision-making, in which it is only the difference between alternatives that drives decision-making [18], while absolute value is deemed irrelevant.

Here, we only model the 'equal alternatives' conditions from the original dataset [17]. The original dataset contained four 'equal alternatives' conditions; for simplicity we median-split these into low-value and high-value conditions. The average reaction time for low-value equal conditions was 784 ms, and 704 ms for high-value conditions. We require that a response is made, but ignore which response is given (i.e., left or right), since for equal alternatives responses should be random, in the absence of a response bias. No bias is reported in the data and analyses.

The two conditions had a mean brightness of .35 (low value) and .55 (high value) on a 0-black to 1-white scale of brightness. We know from psychophysical research [4, 22] that the psychological representation of brightness follows that of the physical stimulus value to a power coefficient $\gamma = 0.5$; hence we assume that participants 'experience' stimuli with intensity $.35^{.5} \approx .6$ and $.55^{.5} \approx .75$.

Our goal is to discover a set of candidate computer programs that achieve a similar performance to that of human participants. We model the decision process by assuming that participants extract samples of evidence from the stimuli and combine them in order to trigger a response. A response is initiated when *model-current* reaches the arbitrary threshold value of 1.2 and the *threshold* operator is selected, setting *model-response* to 1. Note that no single sample of evidence can reach a threshold, since samples have values of .6 and .75 (lower than the 1.2 value needed) – in order to reach a threshold, the system would need to perform operations on the samples (for instance, sum multiple samples) so that a response can be initiated. For simplicity, we set the threshold hyperparameter to an arbitrary value that requires more than one sample of evidence to trigger a choice. However, the value of the threshold could also be evolved as part of the model development system.

As in the lab experiment [17], in our simulations, programs (i.e., our artificial participants) are presented with 360 random repetitions of stimuli for both conditions, for a total of 720 trials.

In this study, we are interested in finding and comparing candidate models when sampling is noiseless vs when sampling is noisy, so that we can compare solutions for a noiseless system (i.e., perfect processing of sensory inputs) to solutions for a system with gradual and noisy accumulation of evidence.

3 Model development system

We developed a simple symbolic cognitive architecture able to manipulate inputs (a numeric representation of stimuli) into outputs (a response). The cognitive architecture and other applications of the model development system are described in depth in previous articles [3, 10, 2, 1] in which this methodology was primarily applied to memory research in cognitive psychology.

There are some task-independent components: a three-slot short-term memory (STM) [5], a clock which records in-task time, a *current* value in working memory, and a *response* value of the system. As explained above, stimuli are represented as numerical values. At the beginning of each trial, all task-independent components are set to zero.

The aim of the system is to make a choice that minimises the discrepancy between reaction time data and the model for the two conditions. A set of *operators* (Table 1) can manipulate the components of the cognitive architecture. For instance, the system can extract samples of evidence from the stimuli, write their values to short-term memory, and perform various other operations that would lead the system to execute a response within the desired time window. These operators, with timings derived from previous research [3, 10] or set arbitrarily, are combined in sequence using genetic programming – these sequences de facto represent a cognitive strategy to solve the problem at hand.

At the beginning of the search, a defined number of strategies are generated at random; in later generations, genetic programming allows the combination and modification of the programs via mechanisms such as *mutation* and *crossover* [9]. For in-depth details on genetic programming, readers should refer to the literature on the topic [9].

Table 1. The operators used in our model development system. Operator timings are as follows: input operators (100 ms), output operators (140 ms), cognitive operators (70 ms), STM operators (50 ms), syntax operators (0 ms), time operators last as long as declared (e.g., wait-25 lasts 25 ms, while-50 lasts 50 ms). Note that, in two separate runs, participants use either *sample* or *noisy-sample*, but not both in the same run.

name	**function**	**type**
access-stm-X	reads the value of slot X in STM. Sets model-current to the value of the slot	STM
compare-X-Y	compares the values of the slots X and Y in STM. If both are equal, it sets model-current to value in slot X, otherwise it leaves model-current unchanged	cognitive
sum-1-2	sets model-current to the sum of the values of the slots 1 and 2 in STM	cognitive
putstm	pushes value in model-current to top of STM	STM
sample	extracts a sample (a reading) from the stimuli and sets model-current to that	input
noisy-sample	extracts a sample (a reading) from the stimuli and sets model-current to that with probability of .75, otherwise it sets model-current to 1	input
threshold	sets model-response to 1 if model-current >1.2 or to 0 if not	output
wait	advances model-clock (in ms): 25 50 100 200 1000 or 15000	time
donothing	does nothing	syntax
dotimesX	repeats a program 2, 3 or 5 times	syntax
prog-x	a sequence of 2, 3 or 4 programs	syntax
if-bigger	if model-current is bigger than 1.2 perform action A otherwise perform action B	syntax

The aim of the genetic programming system is to generate strategies that minimise the discrepancy between the data and the generated solutions – in our case, this means minimising the difference in mean reaction times (for each of the two conditions) and choice between the model and the data. The fitness function includes three components: reaction times for the two conditions, and choice. Recall that 'choice' is modelled as executing a response; which alternative is selected is irrelevant given that alternatives are identical. The reaction times component had a weight of .35 for each of the two conditions, while the choice component had a weight of .30. We chose these values to give an approximate equal weight to each of the three components of the fitness function.

We note here that the operators reported in Table 1 could give rise to sophisticated decision regimes. For instance, participants could extract multiple samples of evidence, write them to short-term memory and then compare or sum them to reach a threshold for a decision. Alternatively, since for equal alternatives information about multiple samples is redundant, participants could simply extract a single sample of evidence and apply a sequence of if-rules in order to arrive at a decision.

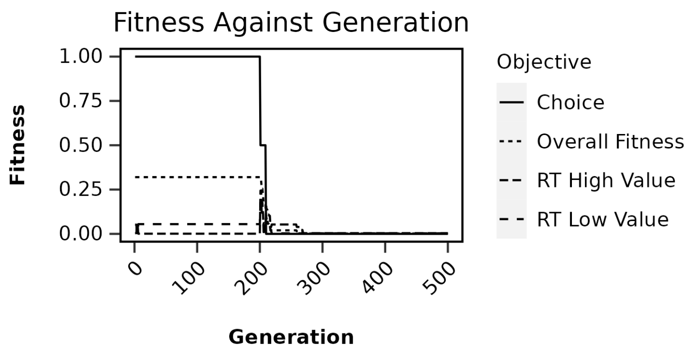

Fig. 2. Fitness against generation. The plot shows the evolution of the overall fitness, and of the sub-components of the fitness function (reaction time for the two conditions, and choice).

After the search phase, the solutions need to be post-processed. Post-processing removes operators that have no effect on the solution (also known as dead-code) [7, 10]. For instance, consider an if-statement such as: 'if 1>2, then action A is performed, otherwise action B is performed'. In this case, action A will never be performed; hence, post-processing would simplify the if-statement into simpler syntax that executes action B only. The amount of dead-code is a function of the number of generations and individuals used during the search phase, and can account for a large percentage of operations included in the solutions generated [7, 10]. Removing dead-code significantly improves the ease of interpretation of the best solution(s), which is important when evaluating whether two models are qualitatively similar, as is often the case [7, 10].

For the search, we evolved 500 individuals for 500 generations. The search was performed using SBCL – the code used for the heuristic search is available by contacting the authors. For the GP implementation, we used mini-gp (https://github.com/jorgetavares/mini-gp) with default values.

4 Results

4.1 Simulation 1: Noiseless sampling

In this simulation we used the *sample* operator, rather the *noisy-sample* operator. Overall, fitness approaches zero (i.e., perfect prediction) after approximately 250 generations, see Figure 2. That is, models after the 250th generation quantitatively explain the data well; as Figure 2 shows, each of the components of the fitness function (reaction times, and choice), and the global fitness function itself, reach a perfect prediction. Models (i.e., individuals) in the final generation have low fitness values, approaching zero. Note the drop in fitness at generation

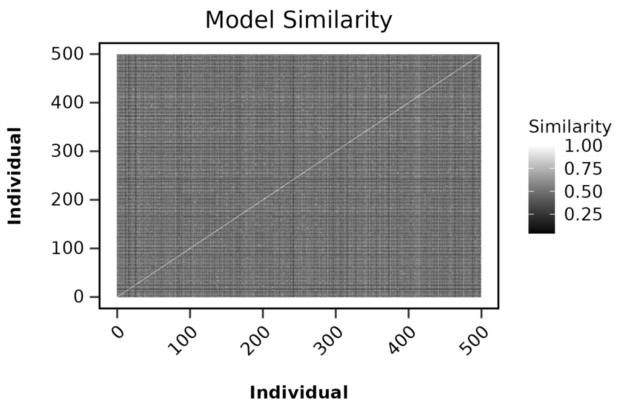

Fig. 3. Model similarity for models in the last generation. Despite all being 'good' enough models (i.e., fitness near zero), the models show low similarity (i.e., they are dissimilar).

200 - models after generation 200 are better than those generated at random at the beginning of the search.

Even though these models have a similar fitness value, they represent different decision-making strategies (see Figure 3). That is, the data are well explained by models that are dissimilar. The similarity between two models is computed by breaking the program down into components and then measuring their similarity using the Jaccard similarity coefficient.

Models include a high proportion of dead-code, an expected feature of GP – program size increases as a function of the number of generations, Figure 4. At generation 200 there is a drop in size that coincides with an increase in fitness - this means that models become better and simpler. However, as common with

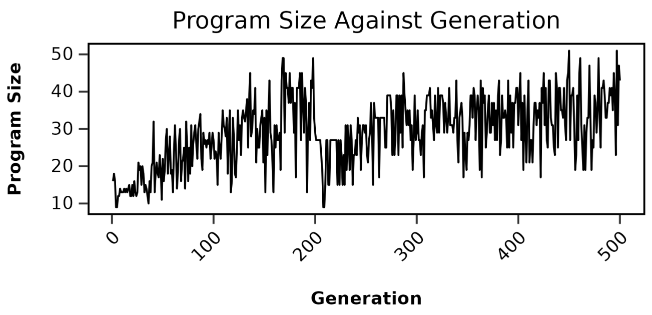

Fig. 4. Increase in program size as a function of generation.

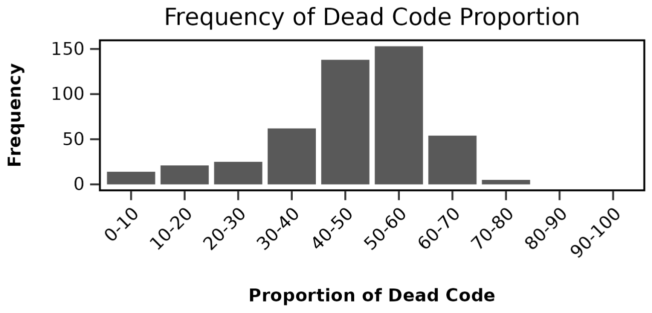

Fig. 5. The frequency of dead-code for models in the final generation (i.e., proportion of a model that is unnecessary).

GP, the programs ‘bloat’ after each generation. As shown in Figure 5, almost half of the models in the final generation consist of about 40-60% of dead-code; few models have a proportion of dead-code higher than 60%.

The best models are defined as models within a tolerance of 0.01 in fitness value from the single best model with the lowest fitness. After post-processing, 66 best models were found; note that they all have identical fitness of 0.002. As in Figure 3, we can show similarity across the best models; Figure 6 shows that the best models have a very similar structure. Note that the fact that models have an identical fitness of 0.002 is no guarantee that the models are syntactically and/or semantically identical; in fact, very different models could achieve a similar fitness.

A representative model from the best solutions is reported in Figure 7. Note that some nodes in the tree are irrelevant (e.g., the first sum-1-2). The model can be interpreted as follows: participants extract a single sample from the stimulus (i.e., they extract a reading from the stimuli and set *model-current* to that value), write that value to each of the three slots in STM, and then sum those values until a threshold is reached. If it is a low value stimulus, they perform an additional operation (*put-stm*), as shown in the right side of the if-statement.

We can further perform multi-dimensional scaling to generate a 2D representation of similarity between models and highlight the presence of clusters of models. Figure 8 shows that there are two distinct classes of models that explain the data equally well. The main difference between the two classes is the use of irrelevant operators, such as operators before the first *sample* or after the *threshold* is reached. Note that the output of those operations is identical while using different operators, and in fact the models have the same overall fitness, and same fitness for each of the objectives of the fitness function. That is, the models are semantically identical while syntactically different.

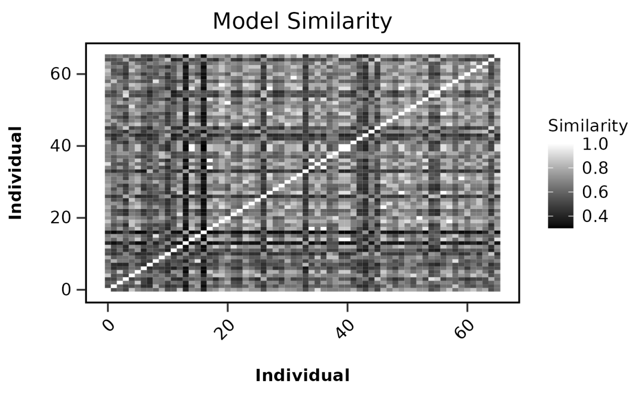

Fig. 6. Model similarity for the 'best' models, after post-processing. 'Best' models are those within a difference of 0.01 from the single best model – note however that in our case, the 66 best models have the same identical fitness of 0.002. The figure shows that these models have a high degree of similarity.

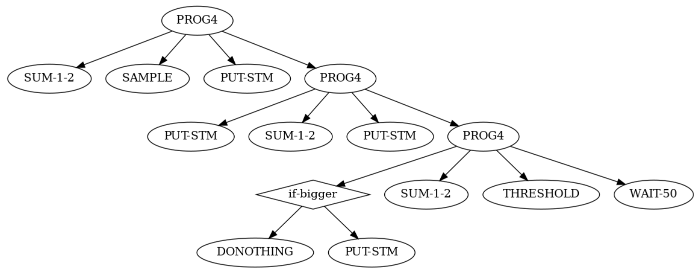

Fig. 7. One of the solutions from the best generation. The strategy is represented using a tree structure; this is standard for solutions from genetic programming. This strategy achieves a fitness of 0.002 (0 is perfect fit), RTs of 780 ms and 710 ms for low and high-value conditions, and a probability of responding of 1.

4.2 Simulation 2: Noisy sampling

In a second separate run of our model development system, we relaxed some of the assumptions of the operators in order to increase the psychological plausibility of the best-estimated models. In particular, we assumed that the operator *sample* is noisy (hence the name *noisy-sample*) and it sets the value of *model-current* to the value of the stimuli only 75% of the time – the remaining 25% of the time the value of *model-current* is set to an arbitrary value of 1. The search estimated a single best tree, reported in Figure 9.

Interestingly, the strategy reported in Figure 9 extracts multiple, separate samples of evidence, writes them to STM and finally sums the values of STM slots 1 and 2 to trigger a response. This model is qualitatively akin to what would

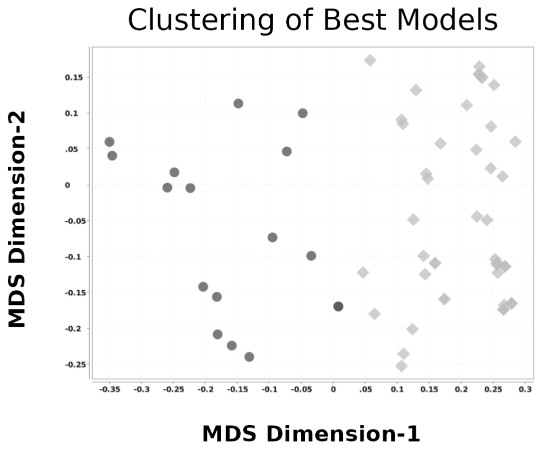

Fig. 8. Clusters for the best models in the final generation. MDS is short for multi-dimensional scaling.

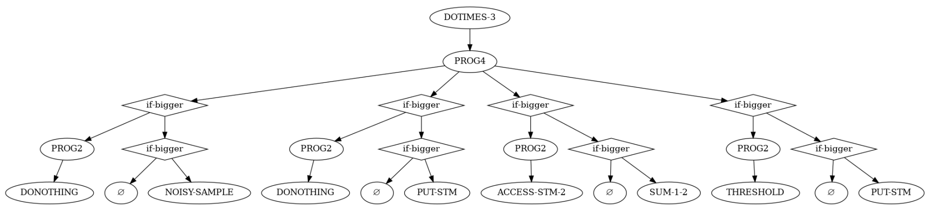

Fig. 9. The single best strategy across all strategies considered by the search algorithm when the operator *sample* is noisy. The 'empty' symbol represents an operator that is never executed, because it is dependent on a value of the model that is never met.

be predicted by an evidence-accumulation architecture [11]. Given the presence of noise, multiple separate samples are extracted and integrated to arrive at a decision – on the other hand, in the fully deterministic, noiseless model, a single sample triggers a decision. Contrary to classical evidence-accumulation models, the strategy reported in Figure 9 requires fewer iterations - in fact, in evidence-accumulation models sampling lasts 1ms and a multitude of samples are extracted to arrive to a decision. Here we find that such a strict requirement may not be necessary.

The methodology presented here allows a wealth of additional analyses. Analyses could focus on aspects such as varying the precision of the sampling operator across runs, estimating the best solution across different sets of operators, highlighting sub-classes of best solutions for different participants rather than considering averaged data or more. These are all exploratory/confirmatory analyses that depend on the specific goal, focus and interest of the researcher. For the remainder of our article, we decided to focus on the implications of this methodology, rather than performing additional analyses.

5 Discussion

We have presented a methodology that allows a researcher, given a dataset and a set of basic building blocks of cognition, to search a wide space of possible strategies that participants may have adopted. To demonstrate the utility of this approach, we applied it to a task in which participants select the brighter of two stimuli. In the experiment, unbeknownst to the participants, conditions of interest are those in which the brightness of the two stimuli are identical.

Our analyses focused on estimating the 'best' strategies in the absence vs presence of noise in the operators. That is, we were interested in evaluating differences in best strategies as a function of the parameterisations in our model development system and underlying assumptions. We found that if we assume a fully noiseless, deterministic strategy, participants extract a single sample of evidence from the stimulus and essentially 'boost' the sample in order to reach a threshold for a response. Intuitively, this is what would be expected in the absence of noise – since a single sample of evidence provides perfect information about the stimuli, it is unnecessary to sample more than once. On the other hand, if sampling is noisy and unreliable, participants extract multiple samples and combine them in order to arrive at a consensus for a response, in an evidence-accumulation fashion. That is, if sampling is noisy, more than one sample is needed to arrive at a decision. In both the noiseless and noisy case, value-sensitivity is implemented via the operator *if-bigger* which, depending on the value of *model-current*, executes one of two operators.

The methodology that we presented here is agnostic concerning applications – while in this case it was applied to a single decision-making task, it has been applied to different domains, such as research on working memory [10, 3]. That is, the same discovery process can be adopted in different domains and applications. Similarly, while for simplicity we applied the methodology to averaged data, this methodology could be adopted to find strategies accounting for individual data. For instance, researchers could use this methodology to infer individual and average strategies during decision-making in order to estimate between-subject similarities in decision strategies, and deviance of individuals from the strategy that explains aggregated results. The same rationale applies to the cognitive architecture and the operators - they can be adjusted by the researcher according to their specific needs.

Even in the case of a typical decision-making task, there are many strategies (virtually an infinite number) that could account for patterns in a dataset; as the complexity of tasks or the number of operators increases, the number of strategies increases exponentially and we believe that this methodology could aid a first exploratory phase in which candidate alternative strategies are discovered. Subsequently, other analyses or ad-hoc experimental work can be devised to compare alternative explanations. For instance, previous work [10] used this methodology for the Delayed Matched to Sample task (an experiment used to study short-term memory) and, over multiple runs, found that three distinct classes of models could explain the data equally well.

The task presented here is arguably a phenomenologically simple task. Despite this, several strategies could explain the simulated data since the decision performance is determined by a large number of underlying factors. The fact that a large number of models can explain any finite set of data is the well known problem of underdetermination of theory from data – especially when both stimulus representations and processes are unknown. However, the general approach in cognitive science is to test predictions of one model/strategy or compare two of them [14]. We believe that in the early phases of theory development, the tool presented here could aid in the search for sensible theories to be considered, beyond those proposed by the researcher's intuition.

This methodology could be useful in multiple cases. Firstly, for researchers who are interested in estimating optimal solutions for a specific problem, given a set of basic cognitive processes. Estimating optimal solutions is hard [21] and often mathematically prohibitive. As a search-based optimization technique, genetic programming could be adopted to estimate optimal strategies for a specific decision problem – this methodology allows researchers to evaluate a large range of solutions and contrast and compare similarities and differences. An interesting avenue for future research is to evaluate how optimal strategies vary when aspects of the system, such as the number or precision of operators, are varied.

A second case in which this methodology could prove useful is to aid researchers in developing hypotheses regarding which strategies could have generated data. For instance, this methodology could support a researchers' hypothesis if it is congruent with the best strategy estimated. On the other hand, if the working hypothesis is not in line with the best strategy estimated, that may lead to a reconsideration of theories.

6 Limitations and future directions

While researchers' intuition may or may not be directly influenced by data when generating theories, the methodology presented here is data-based. If data are of poor quality, the resulting theory will be a theory that explains poor-quality data. To mitigate this risk, this methodology should be applied to large datasets across different types of tasks; if the best strategy *simultaneously* accounts for datasets across domains, conclusions limited by the specifics of the experimental procedure are less likely.

Crucially, this methodology relies on the operators, the building blocks of cognition, to generate a theory. A current limitation of the methodology is that the timings of operators are fixed, either to values from the literature [3] or, for convenience, to arbitrary values. For instance, here we assume that sampling lasts 100 ms. Similarly, all memory operators have an arbitrarily fixed duration of 50 ms. While setting features of cognitive capacities to fixed values is often standard practice, for instance in unified theories of cognition [15], one risk with this approach is to perform computationally expensive model search between models that are unreasonable to begin with, due to incorrect assumptions regarding the timings of operators. However, assumptions, implicit or explicit, permeate every theory of cognitive capacity – indeed, every theory in the empirical sciences. A benefit of this approach is that it makes a wider number of assumptions explicit (e.g., the timing of operators, the cognitive architecture) and it allows the estimation of how changing specific assumptions changes what can be said about cognitive capacities. Our current work is focusing on minimising the risk of fixing certain features to arbitrary values, by varying parameters that we previously kept fixed in our search. This is done by co-evolving crucial features of our system that were hard-coded, such as the timings of operators or the set of operators used as part of the search. That is, hyper-parameters which we have set to arbitrary values in the current article can be evolved as part of the search process. Some hard-coded parameters require extensive changes to the code and architecture in order to be evolved – for instance, the number of slots in short-term memory. Future work will focus on estimating solutions while these hard-coded parameters are either manually varied across runs, or are varied using grid search.

Recently, the use of machine learning has allowed statistical models to be evaluated against large datasets in cognitive science [16]. The methodology described here differs from statistical approaches – here we develop solutions in terms of symbolic, information-processing architectures, and without a dependence on large datasets needed to train statistical models. Crucially, compared to statistical models based on machine learning approaches, solutions are easier to explain and interpret since they are expressed as a symbolic architecture and clearly defined operations performed by the architecture.

Opinions regarding the impact and role of methodologies that semi-automate aspects of scientific discovery vary in the literature. However, one clear benefit of these approaches is to provide an 'existence proof' – given a set of assumptions, we can demonstrate the existence of candidate models for our experimental data. This can then provide insights for the cognitive scientist to develop broader or more understandable theories. The methodology presented here is a potentially useful tool for cognitive scientists when developing theories of cognitive capacities. By generating a wide range of cognitive strategies/models, this methodology aids the cognitive scientist in considering *candidate models*, as opposed to fixating on predictions of a single model or comparisons between two models [14].

Acknowledgments. Funding from the European Research Council (ERC-ADG-835002—GEMS) is gratefully acknowledged.

References

1. Bartlett, L., Pirrone, A., Javed, N., Lane, P.C., Gobet, F.: Genetic programming for developing simple cognitive models. Proceedings of the 45th Annual Meeting of the Cognitive Science Society pp. 2833–2839 (2023)

2. Bartlett, L.K., Pirrone, A., Javed, N., Gobet, F.: Computational scientific discovery in psychology. Perspectives on Psychological Science **18(1)**, 178–189 (2022)
3. Frias-Martinez, E., Gobet, F.: Automatic generation of cognitive theories using genetic programming. Minds and Machines **17**(3), 287–309 (2007)
4. Geisler, W.S.: Sequential ideal-observer analysis of visual discriminations. Psychological Review **96**(2), 267 (1989)
5. Gobet, F., Clarkson, G.: Chunks in expert memory: Evidence for the magical number four... or is it two? Memory **12**(6), 732–747 (2004)
6. Holland, J.H.: Genetic algorithms. Scientific American **267**(1), 66–73 (1992)
7. Javed, N., Gobet, F.: On-the-fly simplification of genetic programming models. In: Proceedings of the 36th annual ACM symposium on applied computing. pp. 464–471 (2021)
8. King, R.D., Rowland, J., Oliver, S.G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L.N., et al.: The automation of science. Science **324**(5923), 85–89 (2009)
9. Koza, J.R.: Genetic programming II: automatic discovery of reusable programs. MIT press (1994)
10. Lane, P.C., Bartlett, L., Javed, N., Pirrone, A., Gobet, F.: Evolving understandable cognitive models. Proceedings of the 20th International Conference on Cognitive Modelling pp. 176–182 (2022)
11. Lee, M.D., Cummins, T.D.: Evidence accumulation in decision making: Unifying the "take the best" and the "rational" models. Psychonomic Bulletin & Review **11**(2), 343–352 (2004)
12. Lieder, F., Krueger, P.M., Griffiths, T.: An automatic method for discovering rational heuristics for risky choice. In: 39th Annual Meeting of the Cognitive Science Society. pp. 742–747 (2017)
13. Marshall, J.A., Reina, A., Hay, C., Dussutour, A., Pirrone, A.: Magnitude-sensitive reaction times reveal non-linear time costs in multi-alternative decision-making. PLoS Computational Biology **18**(10), e1010523 (2022)
14. Meehl, P.E.: Theory-testing in psychology and physics: A methodological paradox. Philosophy of Science **34**(2), 103–115 (1967)
15. Newell, A.: Unified theories of cognition. Harvard University Press (1994)
16. Peterson, J.C., Bourgin, D.D., Agrawal, M., Reichman, D., Griffiths, T.L.: Using large-scale experiments and machine learning to discover theories of human decision-making. Science **372**(6547), 1209–1214 (2021)
17. Pirrone, A., Azab, H., Hayden, B.Y., Stafford, T., Marshall, J.A.: Evidence for the speed–value trade-off: Human and monkey decision making is magnitude sensitive. Decision **5**(2), 129 (2018)
18. Pirrone, A., Reina, A., Stafford, T., Marshall, J.A., Gobet, F.: Magnitude-sensitivity: rethinking decision-making. Trends in Cognitive Sciences **26**(1), 66–80 (2022)
19. Simon, H.A.: Models of discovery, and other topics in the methods of science. Dordrecht, Holland: Reidel (1977)
20. Skirzyński, J., Becker, F., Lieder, F.: Automatic discovery of interpretable planning strategies. Machine Learning **110**, 2641–2683 (2021)
21. Tajima, S., Drugowitsch, J., Pouget, A.: Optimal policy for value-based decision-making. Nature Communications **7**(1), 1–12 (2016)
22. Teodorescu, A.R., Moran, R., Usher, M.: Absolutely relative or relatively absolute: violations of value invariance in human decision making. Psychonomic Bulletin & Review **23**, 22–38 (2016)