Mathematical Formalization and Optimization of an ACT-R Instance-Based Learning Model

Nadia Said[1, 2] (nadia.said@psychologie.uni-heidelberg.de)
Michael Engelhart[2] (michael.engelhart@iwr.uni-heidelberg.de)
Christian Kirches[2] (christian.kirches@iwr.uni-heidelberg.de)
Stefan Körkel[2] (stefan.koerkel@iwr.uni-heidelberg.de)
Daniel V. Holt[1] (daniel.holt@psychologie.uni-heidelberg.de)

[1] Heidelberg University, Department of Psychology, Hauptstr. 47–51, 69117 Heidelberg, Germany
[2] Interdisciplinary Center for Scientific Computing (IWR), Im Neuenheimer Feld 368, 69120 Heidelberg, Germany

Abstract

The performance of cognitive models often depends on the settings of specific model parameters, such as the rate of memory decay or the speed of motor responses. The systematic exploration of a model's parameter space can yield relevant insights into model behavior and can also be used to improve the fit of a model to human data. However, exhaustive parameter space searches quickly run into a combinatorial explosion as the number of parameters investigated increases. Taking an established instance-based learning task as example, we show how simulation using parallel computing and derivative-free optimization methods can be applied to investigate the effects of different parameter settings. We find that both global optimization methods involving genetic algorithms as well as local methods yield satisfactory results in this case. Furthermore, we show how a model implemented in a specific cognitive architecture (ACT-R) can be mathematically reformulated to prepare the application of derivative-based optimization methods which promise further efficiency gains for quantitative analysis.

Keywords: cognitive modeling; ACT-R; instance-based learning; discrete-time systems; derivative-free optimization; parameter identification

Introduction

Most formal models of cognitive processes contain adjustable parameters which moderate model behavior. Exploring the effects of different parameter settings in a cognitive model is important to fully understand its behavior, to identify parameter combinations providing the best fit to human data, and to analyze sensitivity towards parameter variations (see Roberts & Pashler, 2000). In practice, this exploration is still often conducted manually, guided by researcher intuition or sometimes just by trial-and-error. The systematic exploration of a given parameter space is often desirable, but quickly runs into difficulties, as processing time increases exponentially with the number of parameters and the resolution of analysis (the *curse of dimensionality*). Compounding the problem, the computational performance of cognitive models is often comparatively poor as cognitive plausibility usually has priority over computational performance. The development of more efficient methods for parameter space exploration has therefore become an emergent topic in cognitive modeling research (e.g., Best et al., 2009; Gluck, Scheutz, Gunzelmann, Harris, & Kershner, 2007; Lane & Gobet, 2013; Moore, 2011). Complementing existing approaches, we will illustrate how optimization-methods from the field of scientific computing can be applied to parameter search problems. As example we use a model of an implicit learning task, originally implemented in the ACT-R cognitive architecture (Taatgen & Wallach, 2002).

While parallel high-performance computing can improve the speed of parameter space searches, the combinatorial explosion inherent in this task easily exceeds the capacity even of large computing resources (Gluck et al., 2007). Another possibility is to optimize the efficiency of search algorithms. In the current literature, two approaches of this kind can be found. One is to sample the search space selectively, e.g., by Adaptive Mesh Refinement or Regression Trees (Best et al., 2009; Moore, 2011). Areas of the search space with high-information content (e.g., containing discontinuities or non-linear gradients) are sampled more densely, areas with low-information content only sparsely. This strategy allows to preserve most information relevant for modeling purposes while reducing the amount of sampling required. However, instead of approximating the full parameter space, it its sometimes sufficient to find particular points or areas with certain characteristics, e.g., parameter combinations providing the best model fit to empirical data. To reach this goal, derivative-free optimization methods such as genetic algorithms have been employed, which use an evolutionary generate-and-select-strategy to find optimal parameter combinations (e.g., Kase, Ritter, & Schoelles, 2008; Lane & Gobet, 2013). Here, we will illustrate how a cognitive model implemented in a specific cognitive architecture (ACT-R) can be mathematically reformulated towards applying a third general approach, derivative-based optimization, which promises further efficiency gains and additional analytic insights compared to derivative-free optimization.

The Sugar Factory Paradigm

The task used to illustrate this approach is the *Sugar Factory*, a computer-simulated scenario developed by Berry and Broadbent (1984) in order to study how people interact with dynamic systems. In behavorial studies participants are often able to control the *Sugar Factory* system above chance level yet can not verbalize how the system works (Berry & Broadbent, 1984, 1988). This can be explained by assuming that

participants learn to associate particular states of the simulation with specific actions (*instance-based learning*) instead of inducing generalized rules about system behavior (Dienes & Fahey, 1995; Taatgen & Wallach, 2002). The *Sugar Factory* has repeatedly been used in experimental research and the underlying principles of instance-based learning have been shown to apply to more complex situations, ranging from playing backgammon (Sanner, Anderson, Lebiere, & Lovett, 2000) to economic decision making (Gonzalez & Lebiere, 2005) or air-traffic control (Lebiere, Anderson, & Bothell, 2001).

In the *Sugar Factory*, participants are asked to reach a specific sugar production p^* by choosing the number of workers x in each round $j \in [1,N]$. The equation below describes the behavior of the *Sugar Factory*:

$$p_{j+1} = 2 \cdot x_j - p_j + u_r, \tag{1a}$$

$$u_r \in \{-1,0,1\}, \tag{1b}$$

$$p \in \{1,...,12\}, \tag{1c}$$

$$x \in \{1,...,12\}, \tag{1d}$$

where u_r is a random component. If the resulting production is less than 1 or greater than 12, then p is set to 1 or 12 respectively. The goal is to produce 9000 tons of sugar which corresponds to $p = 9$ on each of a number of trials. The initial production value is $p_1 = 6$ and the task is run for 40 rounds. Participants are not informed about the system structure. A sugar output of $p \in [8,10]$ is scored as being *on target*, making it possible to be on target 100 % of the time, despite the random component.

ACT-R Model of the Sugar Factory

Instanced-based learning can be modeled using the declarative memory module of the ACT-R architecture (Anderson et al., 2004). We used an adapted version of the Taatgen and Wallach (2002) model of the *Sugar Factory*, in which instances are represented as memory chunks encoding task state, participant action, and outcome (i.e., current production p_j, number of workers x_j, and new production p_{j+1}).

Each chunk i has an activation value A_i which is computed from three components: the base-level activation B_i, a context component C_i and a noise component $u_{n,ij}$,

$$A_i := B_i + C_i + u_{n,ij}. \tag{2}$$

To retrieve a chunk from memory a retrieval request is made to the declarative module. Only chunks with an activation level above threshold τ are eligible for retrieval.

The base-level activation B_i is calculated from the number n_i of presentations of a chunk i, its the time since its creation L_i and the decay parameter d[1],

$$B_i := \ln\left(\frac{n_i}{1-d}\right) - d \cdot \ln(L_i). \tag{3}$$

[1]Note that we used the ACT-R optimized learning equation.

In this model the context component C_i is determined by the similarity between the numerical values for workers and productions in the retrieval request and corresponding values of the chunks in declarative memory:

$$C_i := P \cdot \sum_k M_{ik}, \tag{4}$$

with M_{ik} as similarity values [2]. The parameter P reflects the amount of weighting given to the similarities.

As the *Sugar Factory* model is very simple, only few production rules are necessary. The steps our model runs through are described below.

1. Start with a number of workers $x = \{7,8,9\}$.
2. Request to retrieve a chunk from memory which matches the current task state and results in a production of $p = 9$.
3. (a) If there is such a chunk and the activation of this chunk is above the threshold τ: perform the action stored in the retrieved chunk, i.e., change the workforce.
   (b) If no chunk reaches the activation threshold τ, perform a random action. If the sugar production is below or above target, then $\{-2,...,2\}$ is added to the current workforce. If the sugar production is on target, then $\{-1,0,1\}$ is added to the current workforce.
4. Create or update a chunk with the information from this round.

The model actions described in rule 1 and rule 3 (b) are based on empirical observations reported in Dienes and Fahey (1995).

Mathematical Reformulation

We start by rewriting the logical relations in the ACT-R model as a recurrence relations system, which is then reformulated using only numerical formulas.

Let N_r be the number of rounds. The general outline of the cognitive process is as follows.

In every round

1. compute the activations of chunks,
2. select the chunk with the highest activation,
3. retrieve information stored in this chunk,
4. perform action,
5. update memory.

[2]The formula for the similarities of numbers was taken from the ACT-R 6.0 Tutorial (2012):

$$M_{ik} := -\frac{|a-b|}{\max(a,b)},$$

For the *Sugar Factory* each chunk represents an instance with i three slots: current production p_j, action (i.e., number of workers x_j), and the new production p_{j+1}. The maximum total number N_c of chunks present in the model can be calculated from the number of values possible for its slots c_{ik}, $k \in \{1,2,3\}$, slot k of chunk i. Feasible values for new workforce (c_{i1}), the current production (c_{i2}) and the new production (c_{i3}) are $\{1,\ldots,12\}$ each. Thus,

$$N_c = 12 \cdot 12 \cdot 12 = 1728. \tag{6}$$

We allocate every possible chunk and set its initial activity to $-M$ where M is sufficiently large.

The lifetime of a chunk consists of the round of its generation t_i, the current round j and a time constant $T = 0.05\,\mathrm{s}$.

The model contains different types of variables:

- *states* including the activation of the chunks and process specific states as the current number of workers and the current production,
- *parameters* τ, d, P and s describing the cognitive properties of the individual participant (the default values in this model are: $\tau = 0$, $d = 0.5$, $P = 10$ and $s = 0.2$)
- *inputs*, containing the cognitive noise u_n, random decisions by the participants $u_w + u_{\text{on}}$ resp. $u_w + u_{\text{off}}$ and system inputs u_r. The sequences of random values are generated a priori as reproducible pseudo-random numbers.

Feasible values for the inputs are: u_{on}, $\in \{-1,0,1\}^{N_r}$ if the production is on target and u_{off}, $\in \{7,8,9\} \times \{-2,...,2\}^{(N_r-1)}$ if the production is off target as well as u_r, $\in \{-1,0,1\}^{N_r}$ for the random vector. All inputs are vectors of length N_r, except $u_{n,ij}$, which is of length $N_r \cdot N_c$. If the target has been reached in round j, i.e. the new production p_{j+1} equals 8, 9, or 10, an indicator R_{j+1} is set to 1. The overall score is computed by summation of R_{j+1}.

This modeling approach leads to a nonlinear recurrence relation system, see Algorithm 1.

In our approach, the logical phrases from the ACT-R formalism are modeled by arg max, $|.|$, max and *if-then* statements. We formulate them using the Heaviside and Delta functions and write the *if-then* statements

$$x = \begin{cases} a, & \text{if} \quad s \geq 0 \\ b, & \text{if} \quad s < 0 \end{cases}, \quad x = \begin{cases} c, & \text{if} \quad t = 0 \\ d, & \text{if} \quad t \neq 0 \end{cases}$$

as

$$x = H(s) \cdot a + (1 - H(s)) \cdot b, \quad x = \delta(t) \cdot c + (1 - \delta(t)) \cdot d.$$

So we substitute $\max(x,y)$ and $\frac{|x-y|}{\max(x,y)}$. To calculate

$$i^* = \arg\max A_i, \quad x_j = \begin{cases} c_{i^*1}, & A_{i^*} \geq \tau \\ u_{w,j}, & A_{i^*} < \tau \end{cases},$$

for $j = 1,\ldots,N_r$ **do**

(1) **for** $i = 1,\ldots,N_c$ **do**

$L_i := (j - t_i) + T;$
$B_i := \ln(n_i/(1-d)) - d \cdot \ln(L_i);$
$M_{i1} := -|p_j - c_{i2}|/\max(p_j, c_{i2});$
$M_{i2} := -|9 - c_{i3}|/\max(9, c_{i3});$
$A_i := B_i + P \cdot (M_{i1} + M_{i2}) + u_{n,ij};$

end

(2) $i^* := \text{argmax}_i A_i;$

(3) $A_{i^*} \geq \tau$?
(i) $A_{i^*} \geq \tau$: $x_j := c_{i^*1}$;
(ii) $A_{i^*} < \tau$: $x_j := u_{w,j}$;

(4) $p_{j+1} := 2 \cdot x_j - p_j + u_{r,j}$;
(i) $p_{j+1} > 12$: $p_{j+1} = 12$;
(ii) $p_{j+1} < 1$: $p_{j+1} = 1$;
(iii) $p_{j+1} = 9$?
(a) $p_{j+1} = 9$: $u_{w,j+1} := u_{w,j} + u_{\text{on},j}$;
(b) $p_{j+1} \neq 9$: $u_{w,j+1} := u_{w,j} + u_{\text{off},j}$;

(5) $u_{w,j+1} > 12$: $u_{w,j+1} = 12$;

(6) $u_{w,j+1} < 1$: $u_{w,j+1} = 1$;

(7) $p_{j+1} \in \{8,...,10\}$?
(i) $p_{j+1} \in \{8,...,10\}$: $R_{j+1} := 1$;
(ii) $p_{j+1} \notin \{8,...,10\}$: $R_{j+1} := 0$;

(8) $\exists i = 1,\ldots,N_c : c_i = (x_j, p_j, p_{j+1})$?
(i) $\exists i$: $n_i := n_i + 1$
(ii) $\nexists i$: $N_c := N_c + 1$;
$c_{N_c} := (x_j, p_j, p_{j+1})$;
$n_{N_c} := 1$;
$t_{N_c} := j$;

end

Algorithm 1: ACT-R algorithm of *Sugar Factory*

we firstly compute $A^* = \max_i A_i$ and then

$$x_j = \sum_{i=1}^{N_c} H(A_i - A^*) \cdot (H(A^* - \tau) \cdot c_{i^*1} + (1 - (A^* - \tau)) \cdot u_{w,j}).$$

In order to compute sensitivities and use derivative-based optimization algorithms at a later stage, we aim for a continuous model formulation. We replace Heaviside and Delta functions by continuous approximate redefinitions

$$H(x) := \frac{1}{\pi} \arctan(h \cdot x) + \frac{1}{2}, \quad \delta(x) := \exp\left(-\frac{x^2}{a^2}\right),$$

with $h = 1000$, $a = 0.01$.

The limits on the production in the *Sugar Factory* are implemented by *if-then* statements. Those rules appear as follows in our mathematical description:

$$p_{j+1} > 12 : p_{j+1} = 12,$$
$$p_{j+1} < 1 : p_{j+1} = 1.$$

In our reformulation those *if-then* statements are smoothened

using the Heaviside function $H()$:

$$\begin{aligned}\tilde{p}_{j+1} &= 2 \cdot x_{j+1} - p_j + u_{rj} \\ p_{j+1} &= H(\tilde{p}_{j+1} - 12) \cdot 12 + \big(1 - H(\tilde{p}_{j+1} - 12)\big) \\ &\quad \cdot \Big(H(1 - \tilde{p}_{j+1}) \cdot 1 + \big(1 - H(1 - \tilde{p}_{j+1})\big) \cdot \tilde{p}_{j+1}\Big).\end{aligned}$$

Simulation

We implemented the reformulation of the model in the *Python* programming language. Simulations were run on a 48-core high-performance server ($4 \cdot 12$-core *AMD Opteron 6176*, non-dedicated) with 256 GB RAM. The maximum runtime per simulation run did not exceed one day. We focused on parameters P and τ, as they have considerable effect on model performance yet no strong empirically based recommendations for default values exist. All other parameters where left at the recommended default values.

For each grid point the simulation was run 100 times with different input vectors, see Figure 1. The activation noise $u_{n,ij}$ was set to zero, as it does not lead to a notable change in mean performance. In a second step, only instances in which a chunk was retrieved were counted as being on target and compared to our previous results (see Figure 1 (b)). Figure 2 shows the sensitivity of results with regard to different initial production values.

In general, all simulation results show a similar pattern in response to parameter variation. Figure 1 (a) shows a characteristic interaction of parameters τ and P, with the *highest performing* parameter combinations located in a wedge-shaped area at the center of the plot and lower performance in both lower left and upper right corners. Considering whether model responses where based on memory retrieval as opposed to random behavior (see 1 (b)) shows that learning occurs in the lower left corner. In contrast, in the upper right corner behavior is almost exclusively driven by random behavior.

Furthermore, Figure 2 shows that the initial starting values of the *Sugar Factory* problem have a notable influence on overall performance, but do not interact in an unpredictable way with parameter settings. The pattern is cognitively plausible, as starting values closer to the actual best control values make system control easier.

Optimization

In order to determine the parameter combination with the highest performance and the best model fit, we applied a number of derivative-free optimization algorithms: Nelder-Mead Simplex (Nelder & Mead, 1965) with explicit support for bound constraints (Box, 1965) and BOBYQA (Powell, 2009) which are both local derivative-free optimization solvers. BOBYQA uses an iteratively constructed quadratic approximation for the objective function. Additionally, we applied ESCH, a modified genetic algorithm (Beyer & Schwefel, 2002) and Controlled Random Search (CRS) with local mutation (Kaelo & Ali, 2006). ESCH and CRS are both heuristic global solvers. CRS starts with a population of random points, and evolves these heuristically, a method comparable to genetic algorithms. All optimization algorithms were applied using the Python interface of NLopt (Johnson, 2014). The stopping criterion for the local solvers was a relative tolerance on the optimization parameters of 0.1. For the heuristic global solvers the stopping criterion was set to a maximum runtime of $960s$.

The objective function for the highest performance was a weighted sum consisting of the mean of the results on target and the standard deviation,

$$a \cdot \frac{1}{n}\sum_{i=1}^{n} R^i + b \cdot \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(R^i - \frac{1}{n}\sum_{i=1}^{n} R^i\right)^2},$$

where $R^i = \sum_j R^i_{j+1}$. R^i_{j+1} is the indicator *on target* in round $j = 1, \ldots, N_r$ for input $i = 1, \ldots, n$, $n = 100$.

Table 1 shows the results for $a = 1$ and $b = 0$ and 100 inputs using multiple start values (see Figure 1 (a)). The local solvers Nelder-Mead and BOBYQA both found the same local maximum ($\tau = -4$, $P = 27$ with objective = 20.15), for ESCH and CRS we successively increased the time limit from 120 to 960 seconds. Table 1 shows the maxima found by the heuristic global solvers after 960 seconds. For $a = 1$ and $b = -1$, all solvers found the same point as a maximum ($\tau \approx -6.5$, $P \approx 30$ with objective ≈ 13.87), except CRS which found a slightly better point ($\tau \approx -8.15$, $P \approx 34.9$ with objective ≈ 14.04).

For optimizing the model fit, the objective function was the root mean square deviation (RMSD) of the model performance and a human reference value $R_{\text{ref}} = 7.9$ taken from the literature (Dienes & Fahey, 1995),

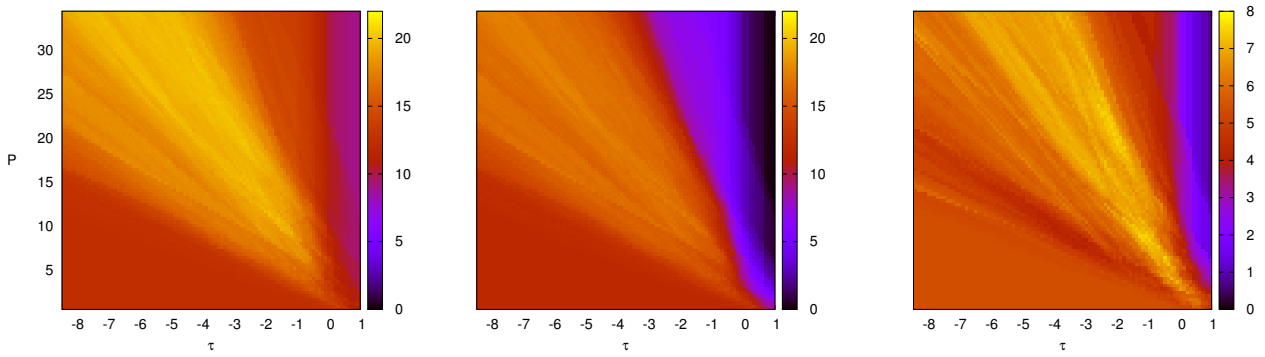
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(R^i - R_{\text{ref}}\right)^2}.$$

Table 2 shows the results for the different solvers again using multiple start values. All solvers found the same point as a maximum ($\tau \approx 0.5$, $P \approx 30$ with objective $= 4.05$). The time limit for ESCH and CRS was set to 5 seconds.

Table 1: Solver comparison, highest performance

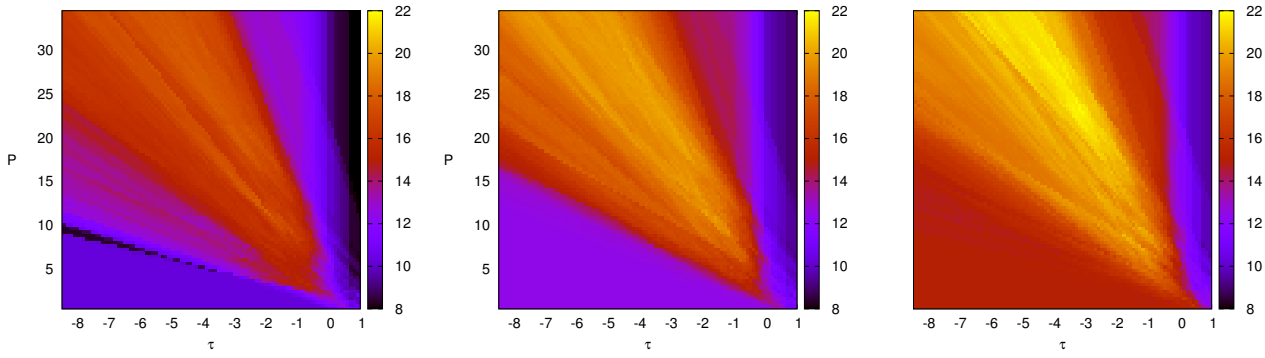
Solver	τ	P	Maximum	#Evaluations
Nelder-Mead	-4	27	20.15	36
BOBYQA	-4	27	20.15	43
ESCH	-3.13	22.36	20.13	863
CRS	-4.21	28.52	20.2	860

The parameter combinations that provide the best fit to average human performance are located at about $\tau = 0.5$ and $P \approx 30$, as indicated by Figure 1 (a) and the results of optimization routines. Interestingly, this combination is far removed from the possible optimal performance, located at parameter values $\tau = -4.21$ and $P = 28.52$. Apparently, consid-



(a) Mean number of rounds on target (b) Mean number, only activated chunks (c) SD, only activated chunks

Figure 1: Rounds on target for 100 inputs over 40 rounds, initial production $p_0 = 6$, medium grid (8256 grid points)



(a) Initial production $p_0 = 1$ (b) Initial production $p_0 = 6$ (standard) (c) Initial production $p_0 = 9$

Figure 2: Rounds on target for 100 inputs over 40 rounds with different initial production values, medium grid (8256 grid points)

Table 2: Solver comparison, best model fit

Solver	τ	P	Maximum	#Evaluations
Nelder-Mead	0.5	28.13	4.05	67
BOBYQA	0.5	27.80	4.05	54
ESCH	0.45	27.88	4.05	6374
CRS	0.48	32.94	4.05	4500

ering even weak memories of previous instances (i.e., a low retrieval threshold τ) is an advantage in this task.

Conclusions

In this work we derive a mathematical formalization of an ACT-R cognitive model to enhance its numerical tractability. The formulation contains both specific parts describing the instances and rules of a particular task and generic parts modeling the declarative memory module of the ACT-R framework. This enables us to apply the approach to a broader range of scenarios. Our formulation transfers the logic-based ACT-R rules to a recurrence relation, which after smoothing represents a differentiable mapping from model parameters and model inputs to the model outputs. In this work we used this formulation for a simulation-based study of the *Sugar Factory* paradigm where we explored the parameter space by parallel computing and computed optima by search-methods and derivative-free approaches. Our next step will be the evaluation of the derivatives of the recurrence relation by using techniques of algorithmic differentiation (Griewank & Walther, 2008) and to apply them in derivative based numerical approaches for sensitivity analysis, parameter estimation and optimum experimental design for efficient model calibration (Körkel, Kostina, Bock, & Schlöder, 2004). Using these methods promises a considerable reduction of computational costs for the quantitative analysis of suitable cognitive process models.

Acknowledgments

This project was supported by the Excellence Initiative of the German Research Council (DFG) at Heidelberg University and the Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences at the Interdisciplinary Center for Scientific Computing (IWR).

References

ACT-R 6.0 Tutorial. (2012). Retrieved from http://act-r.psy.cmu.edu/software/

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*, 1036–1060.

Berry, D. C., & Broadbent, D. E. (1984). On the relationship between task performance and associated verbalizable knowledge. *The Quarterly Journal of Experimental Psychology*, *36*, 209–231.

Berry, D. C., & Broadbent, D. E. (1988). Interactive tasks and the implicit-explicit distinction. *British Journal of Psychology*, *79*, 251–272.

Best, B. J., Furjanic, C., Gerhart, N., Fincham, J., Gluck, K. A., Gunzelmann, G., & Krusmark, M. A. (2009). Adaptive mesh refinement for efficient exploration of cognitive architectures and cognitive models. In *Proceedings of the ninth international conference on cognitive modeling* (paper 252). Manchester, United Kingdom: University of Manchester.

Beyer, H.-G., & Schwefel, H.-P. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing*, *1*, 3–52.

Box, M. J. (1965). A new method of constrained optimization and a comparison with other methods. *The Computer Journal*, *8*, 42–52.

Dienes, Z., & Fahey, R. (1995). Role of specific instances in controlling a dynamic system. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *21*, 848–862.

Gluck, K., Scheutz, M., Gunzelmann, G., Harris, J., & Kershner, J. (2007). Combinatorics meets processing power: Large-scale computational resources for BRIMS. In *Proceedings of the sixteenth conference on behavior representation in modeling and simulation* (pp. 73–83). Orlando, FL: Simulation Interoperability Standards Organization.

Gonzalez, C., & Lebiere, C. (2005). Instance-based cognitive models of decision-making. In D. J. Zizzo & A. Courakis (Eds.), *Transfer of knowledge in economic decision making.* New York: Palgrave McMillan.

Johnson, S. G. (2014). The NLopt nonlinear-optimization package. Retrieved from http://ab-initio.mit.edu/nlop

Kaelo, P., & Ali, M. M. (2006). Some variants of the controlled random search algorithm for global optimization. *Journal of Optimization Theory and Applications*, *130*, 253–264.

Kase, S. E., Ritter, F. E., & Schoelles, M. (2008). From modeler-free individual data fitting to 3-d parametric prediction landscapes: A research expedition. In *Proceedings of the 30th annual conference of the cognitive science society* (pp. 1398–1403). Austin, TX: Cognitive Science Society.

Lane, P. C. R., & Gobet, F. (2013). Evolving non-dominated parameter sets. *Journal of Artificial General Intelligence*, *4*, 358–367.

Lebiere, C., Anderson, J. R., & Bothell, D. (2001). Multitasking and cognitive workload in an ACT-R model of a simplified air traffic control task. In *Proceedings of the tenth conference on computer generated forces and behavior representation.* Norfolk, VA.

Moore, L. R. J. (2011). Cognitive model exploration and optimization: a new challenge for computational science. *Computational and Mathematical Organization Theory*, *17*, 296–313.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*, 308–313.

Powell, M. J. D. (2009). The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge Report DAMTP NA2009/06, University of Cambridge, United Kingdom*.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review*, *107*, 358–367.

Sanner, S., Anderson, J. R., Lebiere, C., & Lovett, M. C. (2000). Achieving efficient and cognitively plausible learning in backgammon. In *Proceedings of the seventeenth international conference on machine learning* (pp. 823–830). Stanford, CA: Morgan Kaufmann.

Taatgen, N. A., & Wallach, D. (2002). Whether skill acquisition is rule or instance based is determined by the structure of the task. *Cognitive Science Quarterly*, *2*, 163–204.