RESEARCH ARTICLE

Applying Mathematical Optimization Methods to an ACT-R Instance-Based Learning Model

Nadia Said[1,2]*, **Michael Engelhart**[2], **Christian Kirches**[2], **Stefan Körkel**[2], **Daniel V. Holt**[1]

1 Institute of Psychology, Heidelberg University, Hauptstr. 47–51, 69117 Heidelberg, Germany,
2 Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany

* nadia.said@psychologie.uni-heidelberg.de

Citation: Said N, Engelhart M, Kirches C, Körkel S, Holt DV (2016) Applying Mathematical Optimization Methods to an ACT-R Instance-Based Learning Model. PLoS ONE 11(7): e0158832. doi:10.1371/journal.pone.0158832

Editor: Hedderik van Rijn, University of Groningen, NETHERLANDS

Received: December 22, 2015

Accepted: June 22, 2016

Published: July 7, 2016

Data Availability Statement: Data are available from https://dx.doi.org/10.6084/m9.figshare.3437468.

Funding: This work was supported by DFG Graduate School 220 (Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences) (http://www.mathcomp.uni-heidelberg.de/). We acknowledge the financial support of the Deutsche Forschungsgemeinschaft and Ruprecht-Karls-Universität Heidelberg within the funding program Open Access Publishing. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Abstract

Computational models of cognition provide an interface to connect advanced mathematical tools and methods to empirically supported theories of behavior in psychology, cognitive science, and neuroscience. In this article, we consider a computational model of instance-based learning, implemented in the ACT-R cognitive architecture. We propose an approach for obtaining mathematical reformulations of such cognitive models that improve their computational tractability. For the well-established *Sugar Factory* dynamic decision making task, we conduct a simulation study to analyze central model parameters. We show how mathematical optimization techniques can be applied to efficiently identify optimal parameter values with respect to different optimization goals. Beyond these methodological contributions, our analysis reveals the sensitivity of this particular task with respect to initial settings and yields new insights into how average human performance deviates from potential optimal performance. We conclude by discussing possible extensions of our approach as well as future steps towards applying more powerful derivative-based optimization methods.

Introduction

Modern cognitive architectures, such as ACT-R [1], allow researchers to construct computational models of behavior that adequately reflect the complexity of human cognition while still being fully formalized. Cognitive architectures are typically based on empirical behavioral studies and neurophysiological research. Using a cognitive model of decision making, it becomes possible to answer questions such as "how does a typical decision maker behave in a particular situation" or "what can be expected, in the best or worst case, from a decision maker".

Cognitive *models* usually focus on specific cognitive phenomena, while cognitive *architectures* are concerned with the general structure of the cognitive system across different tasks. Different types of cognitive architectures based on symbolic, connectionist, or hybrid frameworks exist, such as Soar [2, 3], Leabra [4], Nengo [5], and ACT-R [1]. The increasing

Competing Interests: The authors have declared that no competing interests exist.

availability and use of formal models in the behavioral sciences provides a foundation for applying advanced mathematical tools and methods [6, 7].

Parameter Identification The behavior exhibited by a cognitive model typically depends on multiple *model parameters*, e.g., the rate of memory decay or the amount of cognitive noise. Understanding the parameter space of a given cognitive model and efficiently estimating parameter values that best match an expected or measured behavior is a central task in cognitive modeling. This task is made difficult by the large number of function evaluations required, and by the necessary computational complexity of relevant models. Exploring the effects of different parameter values 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 [8]. In practice, for cognitive models this is often still conducted manually, guided by a researcher's intuition or simply by trial-and-error.

Developing techniques for efficient parameter space exploration and parameter estimation is still a relatively new research area in cognitive modeling, and only a few systematic approaches have been described in the literature to date, e.g. [9–13]. Systematic exploration of a model's 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 (*curse of dimensionality*). While parallel high-performance computing can improve the speed of parameter space searches to some extent, this combinatorial explosion easily exceeds the capacity even of large computing resources [10].

Another possibility is to improve the efficiency of search algorithms. One approach is to sample the search space selectively, for example using adaptive mesh refinement or regression trees [9, 13], where regions of the search space with high-information content are sampled more densely. This strategy allows to preserve most of the information relevant for modeling purposes, while reducing the number of samples required.

Instead of attempting to approximate the full parameter space, it is sometimes sufficient to identify particular points or areas with certain characteristics, e.g., parameter combinations that provide the best model fit to empirical data. To reach this goal, *heuristic optimization methods* such as genetic algorithms have been employed, which use an evolutionary generate-and-select-strategy to find optimal parameter combinations [11, 12]. These heuristic approaches, however, not only require drastically higher computational resources with increasing number of dimensions, but also usually do not come with a proof of optimality of the obtained parameter estimate. Using *mathematical optimization methods*, these issues may partially be avoided by taking information found in (approximations of) first order derivatives of model and objective function into account. This, however, requires an appropriate mathematical reformulation of the model.

Contribution This article proposes an optimization-based approach for evaluating the behavior of a cognitive model of instance-based learning implemented in the ACT-R cognitive architecture. We propose to rewrite the model in terms of mathematically tractable expressions and to apply methods from mathematical programming in order to identify parameter values that are optimal with respect to a prescribed criterion. Our approach is generic in the sense that it may be applied to any ACT-R model based on declarative working memory, and may in principle be automated. Extensions to a much wider class of ACT-R models are possible.

To illustrate our approach, we work with an ACT-R model of the *Sugar Factory* dynamic decision making task [14–16]. We first conduct a simulation study for the analysis of two central model parameters. We then show how to address two common optimization problems: Firstly, the identification of parameter values for the best model fit to human reference values, and, secondly, the determination of parameter values that maximize the performance score. In addition to heuristic optimization methods, we apply derivative-based methods that, given an initial guess, construct descent paths to a minimizer instead of searching the entire parameter space, thereby improving computational efficiency.

Beyond these methodological contributions, our analysis allows us to quantify to what extent performance in the *Sugar Factory* task depends on initial conditions, which is informative for the experimental use of the *Sugar Factory*. Our results furthermore yield new insights into how average human performance deviates from potential optimal performance.

Mathematical Optimization

Our aim is the application of mathematical optimization methods to a cognitive model of decision making to optimize its fit to human behavioral data and to identify conditions of optimal performance. To this end, we formulate mathematical optimization problems and choose appropriate mathematical optimization methods to solve them efficiently.

Optimization Targets Our dynamic decision making task setting is round-based, where we denote rounds by $j = 1, \ldots, N_r$. For a given parameter vector θ, the model behavior may also depend on a pseudo-random sequence of inputs. Then, evaluations take place over repetitions $i = 1, \ldots, n$ with differing realizations of the pseudo-random input sequence. We consider two optimization tasks with respect to the model of the cognitive process:

1. **Fit optimization**. We determine a parameter vector $\theta \in \mathbb{R}^{n_\theta}$ that gives rise to best model fit to human reference values. For optimizing the model fit, the objective function is the root mean square deviation (RMSD) of the model performance and a human reference value R_{ref},

$$\min_{\theta} \quad \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(R^i(\theta) - R_{\text{ref}}\right)^2}, \tag{1}$$

where $R^i(\theta) = \sum_{j=1}^{N_r} R^i_{j+1}(\theta)$. Herein, $R^i_{j+1}(\theta)$ denotes a zero-one indicator that the process was *on target*, i.e. a certain prescribed goal was reached, in repetition i after round j and for model parameters θ.

2. **Performance optimization**. We determine a parameter vector $\theta \in \mathbb{R}^{n_\theta}$ with best score. The objective function for the best score is a weighted sum consisting of the performance criterion, here the mean of the rounds on target, and its standard deviation,

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

Constants $a, b \in \mathbb{R}$ are weighting factors.

Mathematical Reformulation of the ACT-R Model

The cognitive architecture used in this article is ACT-R, a computational framework for modeling higher level cognition [1]. ACT-R consists of three main components: modules, buffers, and a pattern matcher, which are associated with distinct cortical regions. A central production system coordinates the behavior of these modules, see Fig 1. In several functional magnetic resonance imaging (fMRI) studies, Anderson et al. (2007) [17] identified a number of brain regions corresponding to modules in ACT-R, supporting the structure of the architecture.

One important feature of ACT-R is that it combines the symbolic structure of cognition, i.e., how knowledge is encoded as high-level structures, with subsymbolic processes which represent an "[. . .] abstract characterization of the role of neural computation in making that

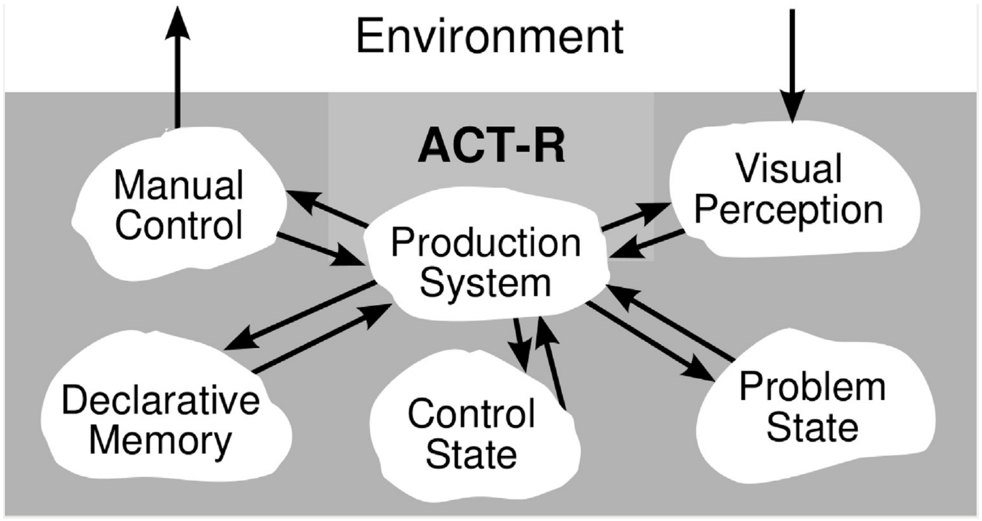

Fig 1. Connection of modules in ACT-R 5.0 [18].

doi:10.1371/journal.pone.0158832.g001

knowledge available," [17]. As an example, instances of symbolic declarative knowledge (e.g., "The number 2 is followed by the number 3."), called chunks, are stored in declarative memory. On the subsymbolic level, an activation value is associated with each chunk and determines whether the information is accessible in a certain situation (e.g., when counting). In contrast to purely connectionist models, in which specific cognitive phenomena emerge from interconnected networks of simple units [19], ACT-R operates on different levels of abstraction in order to achieve a representation of how the components of the mind interact.

Mathematical Description of the Declarative Memory Module

The proposed reformulation of the *Sugar Factory* task (see below) includes a generic representation of a central part of the ACT-R cognitive architecture, the *declarative memory module*. Our approach can therefore be applied in a straightforward manner to other cognitive tasks that rely on this cognitive module.

A single element of declarative knowledge is called a *chunk*, stored in the *declarative memory* module of the ACT-R architecture. A chunk, see Fig 2, is defined by its chunk type and contains a number of *slots* c_{ik} that hold information. Each chunk also has an *activation value* A_i that reflects the usefulness of the stored information for the specific situation at hand [17].

Definition 1 (Chunk and Declarative Memory) *A chunk is a tuple* $(c_{i1}, \ldots, c_{ik}, A_i) \in I_1 \times \ldots \times I_z \times \mathbb{R}$. *The declarative memory is an ordered list* $\mathcal{M}$ *of chunk tuples indexed by consecutive ascending natural numbers.*

The current context and all past experience influence the activation value A_i of a chunk 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},

$$A_i := B_i + C_i + u^{\mathrm{n}}_{ij}. \tag{2}$$

The base-level activation B_i is calculated from the number n_i of presentations of a chunk, its lifetime L_i, and the decay parameter d,

$$B_i := \ln(n_i/(1-d)) - d\ln(L_i). \tag{3}$$

A chunk is said to have been *presented* when it first enters declarative memory, $n_i = 1$, or when

```
(p start
  =goal>
    isa game-state
    state start
    sugar production =p
  ==>
  =goal>
    state retrieving
  +retrieval>
    isa learned-info
    sugar production =p
    newproduction 9
)
```

```
goal
isa game-state
state start
workers 3
sugar production 7
```

```
learned-info12
isa learned-info
workers 8
sugar production 7
newproduction 9
```

Fig 2. Examples of an ACT-R production rule (left) and of ACT-R chunks stored in declarative memory (right).

doi:10.1371/journal.pone.0158832.g002

it is recalled or encountered again, $n_i > 1$. With each additional presentation of a chunk, the base-level activation B_i increases [17]. The lifetime L_i is the time since creation of the chunk. In the case of the *Sugar Factory* implementation, L_i is calculated from the round t_i of a chunk's creation, the current simulation round j, and a time constant $T = 0.05$ s,

$$L_i := (j - t_i) + T. \tag{4}$$

When faced with the current situation in turn j, a *retrieval request* is be made to retrieve a chunk from declarative memory that best matches the current situation. Then, from the subset of chunks that are a satisfactory match of request (p_j, p^*), comprised of a *situation* p_j and a *desired target* p^*, the one with the highest activation value is placed into the *retrieval buffer*.

Definition 2 (Retrieval of a Chunk) *Given a retrieval request* (p_j, p^*), *the index of the chunk retrieved from declarative memory is*

$$i^* = \underset{i}{\operatorname{argmax}} \Big\{ A_i(p_j, p^*) \geq \tau \Big\}. \tag{5}$$

The retrieval threshold τ *defines the minimum activation threshold for a chunk to be retrievable at all. The retrieved chunk is*

$$(c_1^*, \ldots, c_k^*, A^*) = (c_{i1^*}, \ldots, c_{ik^*}, A_{i^*}(p_j, p^*)). \tag{6}$$

The context component $C_i(p_j, p^*)$ contributes a similarity part that reflects the similarity between the slot values (p_j, p^*) of a retrieval request and the slot values $(c_{i1}, \ldots, c_{ik})$ of any chunks in declarative memory. It is not required that the slots of the chunk have exactly the same values as specified in the retrieval request, but C_i increases if their similarity is high. This mechanism is called *partial matching*,

$$C_i(p_j, p^*) := P \cdot \sum_l M_{i,l}, \tag{7}$$

wherein the parameter P reflects the amount of weighting given to the similarities, and the similarity measures $M_{i,\,l}$ are calculated as

$$M_{i,l}(a,b) := -|a-b| / \max(a,b). \tag{8}$$

Maximum similarity between two values is represented by $M_{i,\,l} := 0$ and maximum dissimilarity by $M_{i,\,l} := -1$.

Finally, the noise value u_{ij}^{n} added to the activation consists of two subcomponents: a transient component u_{ij}^{n}, which is computed each time a retrieval request is made, and a permanent component, which is only generated once for each chunk. The transient component is usually sufficient for modeling. To generate the value of the transient noise component a logistic distribution with $\mu = 0$ and noise parameter $s \approx 0.2$ is used [20].

Mathematical Description of the Task and Production Rules

Different modules of ACT-R interact with each other through a production system. The steps the *Sugar Factory* model runs through are described below. In every round:

1. compute the activations of memory chunks;
2. select the chunk with the highest activation regarding a particular recall request;
3. if there is such a chunk and the activation of this chunk is above the threshold τ: make choice stored in chunk b^1;
4. if there is no such chunk or the activation of the chunk is lower than the threshold τ: make random choice b^2;
5. update the *Sugar Factory* system state;
6. create a new memory chunk or merge information acquired with an existing chunk.

Both general cognitive processes and production rules for simulating performance in a specific task are described in ACT-R by a system involving logical relations. In contrast, we aim to formulate a continuously differentiable mathematical model that is a suitable input to mathematical optimization methods. The relevant logical phrases from the ACT-R formalism are argmax, $|\cdot|$ and max, and conditional *if-then* statements. We propose formulations for all three components based on the Heaviside and Delta functions $H(x)$ and $\delta(x)$:

$$H(x) = \begin{cases} 1 & \text{if} \quad x \geq 0, \\ 0 & \text{if} \quad x < 0, \end{cases} \qquad \delta(x) = \begin{cases} 1 & \text{if} \quad x = 0, \\ 0 & \text{if} \quad x \neq 0. \end{cases} \tag{9}$$

Formulation of *if-then* statements. We write *if-then* statements

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

as $x(s) = H(s) \cdot a + (1 - H(s)) \cdot b$ and $y(t) = \delta(t) \cdot c + (1 - \delta(t)) \cdot d$.

Formulation of max **and** $|\cdot|$**.** We substitute max and $|\cdot|$ by

$$\begin{aligned} \max(x,y) &= H(x-y) \cdot x + (1 - H(x-y)) \cdot y, \\ \frac{|x-y|}{\max(x,y)} &= H(x-y)\frac{x-y}{x} + (1 - H(x-y))\frac{y-x}{y}. \end{aligned}$$

Formulation of argmax. To evaluate the statement

$$i^* = \underset{1\le i\le n}{\operatorname{argmax}}\{A_i\}, \quad x_j(i^*) = \begin{cases} b^1 & \text{if} \quad A_{i^*} \ge \tau, \\ b^2 & \text{if} \quad A_{i^*} < \tau, \end{cases}$$

we first compute $A^* = \max_i\{A_i\}$, and then let

$$x_j(i^*) = \sum_{i=1}^{n} H(A_i - A^*) \cdot (H(A^* - \tau) \cdot b^1 + (1 - (A^* - \tau)) \cdot b^2).$$

In order to obtain a continuously differentiable formulation, we then replace Heaviside and Delta functions by continuous approximations,

$$\begin{aligned} H(x) &:= \tfrac{1}{\pi} \arctan(hx) + \tfrac{1}{2}, \\ \delta(x) &:= \exp(-x^2/a^2), \end{aligned}$$

with, e.g., $h = 10.0$, $a = 0.01$.

The Sugar Factory Task

In this article, we investigate an ACT-R model of the *Sugar Factory* decision making task [16]. The *Sugar Factory* is a turn-based task realized as a computer-based simulation, which was developed by [14] to study how people learn to control intransparent dynamic systems. Instead of inducing explicit rules for controlling the system, participants seem to store instances of situation-response pairs in memory, which are retrieved when a similar situation occurs, see [15, 16]. This cognitive mechanism is known as *instance-based learning* (IBL), cf. [21], and has been shown to play an important role in dynamic decision making, e.g., [22]. IBL has been implemented successfully in several cognitive models based on the ACT-R architecture as reported in [16, 22].

In the *Sugar Factory* task, participants control a simulated sugar factory. They are asked to reach and maintain a specific level of sugar production p^* over turns $j = 1, \ldots, N_r$ by repeatedly changing the number of workers x_j employed at the factory. The initial production is $p_1 = 6$. In every round j, the goal is to reach a production of $p_j = p^* = 9$, i.e., to produce 9,000 metric tons of sugar. The following equation describes the behavior of the *Sugar Factory* task.

Definition 3 (Sugar Factory Simulation Problem) *The sugar production rate before turn* $j = 1$ *is* p_1 *and the rate* p_{j+1} *after turn* $j = 1, \ldots, N_r$ *is given by*

$$p_{j+1}(x) = \left(2 \cdot x_j - p_j(x) + u_j^r\right)_{[1,12]}, \tag{10}$$

where $x_j \in \{1, \ldots, 12\}$ *is a sequence of inputs,* u_j^r *is uniformly distributed random variable from* $\{-1, 0, 1\}$, *and* $(y)_{[a,\,b]} = \max(a, \min(b, y))$ *denotes the clipping of the argument value y to the range* $[a, b]$.

Participants are initially unaware of the relationship Eq (10) between workers and sugar production, and are not informed about their results being evaluated in this way.

To measure the performance of a participant in the *Sugar Factory*, we define the following score.

Definition 4 (Sugar Factory Score Function) *The sugar factory score function is*

$$R = \sum_{j=1}^{N_r} R_{j+1} = \sum_{j=1}^{N_r} \chi\left\{ |p_{j+1}(x) - p^*| \le 1 \right\}$$

with $p^* = 9$, *i.e., the score counts the number of rounds where the sugar production rate is on target.*

To account for the randomness in u_j^r and to make it possible for participants to be on target 100% of the time, a sugar production of $p_j \in [8, 10]$ is also scored as being on target.

Human Performance in the Sugar Factory

It has repeatedly been found that human participants are able to control the simulated system above chance level but perform far from the theoretical optimum in this task [14, 15]. Moreover, even successful participants are often unable to verbally describe the structure of the system. This is in line with the assumptions of instance-based learning as a cognitive mechanism which does not require the abstraction of formal rules or relations. Surprisingly, even when the structure of the underlying system is made explicit to participants, they are generally not able to improve their performance [14].

Analyzing individual decision behavior, [15] found that up to 86% of the initial ten choices $x_1, \ldots, x_{10}$ made by participants can be explained by the following rules, which form the basis for the cognitive model further below:

- Initially, a workforce of $x_1 = 7, 8$, or 9 is chosen;
- If the sugar production is below or above target, $p_j < 8$ or $p_j > 10$, then $x_j = x_{j-1} + u_j^{\text{off}}$, where $u_j^{\text{off}} \in \{-2, \ldots, 2\}$ is added to the current workforce;
- If the sugar production is on target, $8 \le p_j \le 10$, then $u_j^{\text{on}} \in \{-1, \ldots, 1\}$ is added to the current workforce.

As an example and to demonstrate the mathematical description of the production rules, the limits on the sugar production rates in the *Sugar Factory* are implemented by *if-then* statements. These rules appear as follows in our mathematical description:

$$\text{if } p_{j+1} > 12 \text{ then } p_{j+1} = 12, \quad \text{if } p_{j+1} < 1 \text{ then } p_{j+1} = 1.$$

In our reformulation, these *if-then* statements are smoothened using the Heaviside function H:

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

Nonlinear Recurrence Model

For the *Sugar Factory* problem, let N_r be the number of rounds. Each chunk i has three slots (c_{i1}, c_{i2}, c_{i3}), where c_{i1} holds the information about the new workforce, the value c_{i2} represents the current production and c_{i3} is the new production calculated from c_{i1} and c_{i2}. The maximum number N_c of chunks can be calculated from the number of values c_{ik} possible for slot $k \in \{1, 2, 3\}$ of chunk i. Feasible values for new workforce c_{i1}, current production c_{i2}, and new production c_{i3} are $\{1, \ldots, 12\}$ each. Thus, $N_c = 12 \cdot 12 \cdot 12 = 1{,}728$. We allocate every possible chunk and set its initial activity to a sufficiently small negative value $-M$ to make sure that it is possible to activate it only after information has been stored in the slots of the chunk.

The mathematical model contains different types of variables:

- *states* including the activation A_i of the chunks, the current number of workers x_j, and the current sugar production rate p_j in the *Sugar Factory*,

- *parameters* $\theta = (\tau, d, P)$ and s describing selected cognitive properties of the individual participant, and
- *pseudo-random vectors*, containing the cognitive noise u^{n}, random decisions by the participants $u_w + u_j^{\mathrm{on}}$ resp. $u_w + u_j^{\mathrm{off}}$ and system inputs u^{r}. The sequences of random values are generated a priori as reproducible pseudo-random numbers.

All inputs are vectors of length N_r, except u_{ij}^{n}, which is of length $N_r \cdot N_c$. The value R_{j+1} is used as an indicator whether the participant has reached the target in round j, i.e., whether the new sugar production p_{j+1} equals 8, 9, or 10. The overall score R^i is computed by summation over all R_{j+1},

$$R^i = \sum_{j=1}^{N_r} R_{j+1}^i.$$

with R_{j+1}^i as the indicator *on target* in round $j = 1, \ldots, N_r$ for input $i = 1, \ldots, n$. This modeling approach leads to a system of nonlinear recurrence relations as shown in Box 1.

Box 1. Algorithm 1: Mathematical formulation of the ACT-R model of the *Sugar Factory*.

```
for $j = 1, \ldots, N_r$ do
    (1) for $i = 1, \ldots, N_c$ do
            $L_i := (j - t_i) + T$;
            $B_i := \ln(n_i/(1-d)) - d \cdot \ln(L_i)$;
            $M_{i1} := -|p_j - c_{i2}|/\max(p_j, c_{i2})$;
            $M_{i2} := -|9 - c_{i3}|/\max(9, c_{i3})$;
            $A_i := B_i + P \cdot (M_{i1} + M_{i2}) + u_{ij}^{\mathrm{n}}$;
        end
    (2) $i^* := \mathrm{argmax}_i\{A_i\}$;
    (3) $A_{i^*} \geq \tau$?
            (i) if $A_{i^*} \geq \tau$ then $x_j := c_{i^*1}$;
            (ii) else        $x_j := u_{w,j}$;
    (4) $p_{j+1} := 2 \cdot x_j - p_j + u_j^{\mathrm{r}}$;
            (i) if $p_{j+1} > 12$ then $p_{j+1} = 12$;
            (ii) if $p_{j+1} < 1$ then $p_{j+1} = 1$;
            (iii) $p_{j+1} = 9$?
                (a) if $p_{j+1} = 9$ then   $u_{w,j+1} := u_{w,j} + u_j^{\mathrm{on}}$;
                (b) else                  $u_{w,j+1} := u_{w,j} + u_j^{\mathrm{off}}$;
    (5) if $u_{w,j+1} > 12$ then $u_{w,j+1} = 12$;
    (6) if $u_{w,j+1} < 1$ then $u_{w,j+1} = 1$;
    (7) $p_{j+1} \in \{8, \ldots, 10\}$?
            (i) if $p_{j+1} \in \{8, \ldots, 10\}$ then   $R_{j+1} := 1$;
            (ii) else                              $R_{j+1} := 0$;
    (8) $\exists i = 1, \ldots, N_c$: $c_i = (x_j, p_j, p_{j+1})$?
            (i) if $\exists i$ then  $n_i := n_i + 1$
            (ii) else       $N_c := N_c + 1$;
                            $c_{N_c} := (x_j, p_j, p_{j+1})$;
                            $n_{N_c} := 1$;
                            $t_{N_c} := j$;
end
```

Properties of the Model and Choice of Optimization Methods

We have implemented the mathematical reformulation of the *Sugar Factory* model in *Python*. Our implementation is modular and object-oriented, with model-specific components encapsulated in a problem class that is easy to substitute in order to transfer it to similar tasks. In this section, we report simulation results obtained using this implementation.

Simulation and Sensitivity Analysis

Computations of the simulations were run on a non-dedicated 48-core machine (4 × 12-core *AMD Opteron 6176*) with 256 GB RAM. To give an impression of the computational effort involved, the maximum runtime for searching even the finest grid investigated did not exceed one day. There were no noticeable differences between the runtimes of our *Python* and the standard ACT-R implementation.

We focused on an analysis of the parameters P (weighting given to similarities) and τ (retrieval threshold), which have considerable effect on the achieved score. The decay rate was set to its default value $d = 0.5$. The activation noise u_{ij}^{n} was set to zero as it does not lead to a noticeable change of the mean score. We describe the random components u^{on}, u^{off} (number of workers added to the current workforce depending on whether the sugar production is on or off target), and u^{r} (random variable added to the sugar production) by pseudo-random input sequences.

Fig 3 shows simulation results for one fixed input (i.e., for every parameter combination (τ, P) the same input sequences u^{on}, u^{off}, and u^{r} were used). Parameter ranges are $\tau \in [-8.50, 1.00]$ with step size $\Delta\tau = 0.05$, and $P \in [0.5, 35.0]$ with step size $\Delta P = 0.1$, which results in a total of 66,086 sampling points. There are certain parameter combinations for which the model is on target about [87.5]% of the time ($\tau \in [-3.3, -0.85]$ and $P \in [8.4, 23.7]$) and others where the score drops to less than 25%. The structure of the plots, especially in the area of the *best learners*, strongly depends on the pseudo-random inputs, compare Figs 3 and 4. In the latter, the *best learners* are on target no more than 50% of the time, the score drops to 10% near the edge. This difference in performance arises from the effect of early random decisions of the model (determined by the pseudo-random input sequences u_0^{on} and u_0^{off}) as memory chunks *on target*, which are important for guiding model behavior, are created early by lucky random choices.

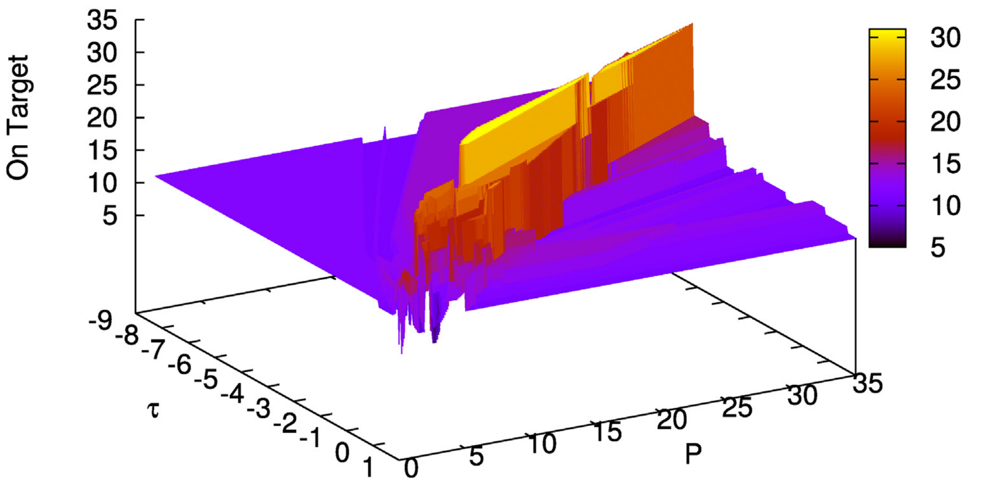

Fig 3. Rounds on target for input$_0$ (= u_0^{on}, u_0^{off}, u_0^{r}) over 40 rounds on fine parameter grid with 66,086 grid points. With $u_0^{on} \in \{-1, \ldots, 1\}$, $u_0^{off} \in \{-2, \ldots, 2\}$, and $u_0^{r} \in \{-1, 0, 1\}$.

doi:10.1371/journal.pone.0158832.g003

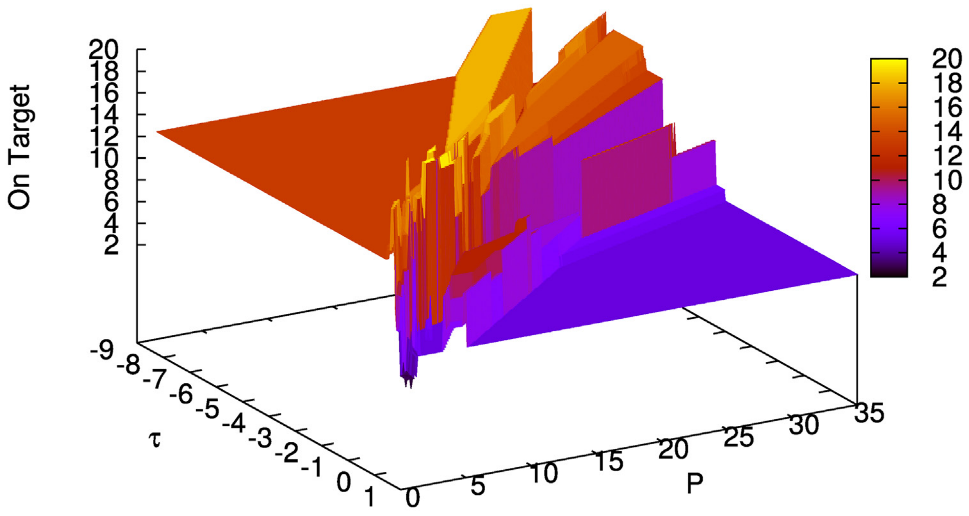

Fig 4. Rounds on target for input$_1$ (= u_1^{on}, u_1^{off}, u_1^{r}) over 40 rounds on fine parameter grid with 66,086 grid points. With $u_0^{on} \in \{-1, \ldots, 1\}$, $u_0^{off} \in \{-2, \ldots, 2\}$, and $u_0^{r} \in \{-1, 0, 1\}$.

doi:10.1371/journal.pone.0158832.g004

Hence, we conducted a second simulation in which the input sequences were varied pseudo-randomly. Fig 5 (left) shows the mean performance for 100 different pseudo-random sequences. Not only does the total number of rounds on target differ compared to the single inputs, but the area of parameter combinations that yield good results is also much broader. To investigate to what extent performance depends on learning, only trials in which a previously learned chunk was retrieved were counted in Fig 5 (center), using the same 100 pseudo-random sequences. Compared to Fig 5 (left), there is a drop of the score in the upper right quarter of Fig 5 (center, standard deviations right). This pattern reveals that effective learning only occurred in the lower left corner (i.e., chunks are recalled) while in the upper right corner high values of P and *tau* impede recall and behavior is driven mostly by random choices. As detailed in the *Cognitive Interpretation* further below, loose criteria for recall (i.e., a low recall threshold and low required similarity of chunks) seem to be beneficial in this task.

In a further simulation, we investigated the sensitivity of the scores with respect to different initial sugar production values. The default value used in experiments is $p_1 = 6$. Results for an

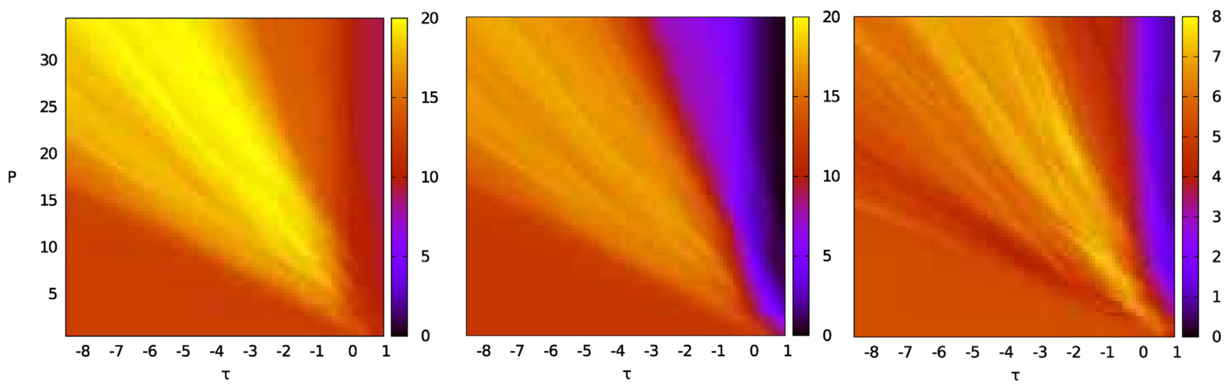

Fig 5. Mean value and standard deviation of rounds on target for 100 different inputs over 40 rounds. Initial sugar production rate $p_1 = 6$, medium parameter grid with 8,256 grid points. Left: Mean value of rounds on target. Center: Mean value, activated chunks only. Right: Standard dev., activated chunks only.

doi:10.1371/journal.pone.0158832.g005

initial value of $p_1 = 9$ show, compared to the default value $p_1 = 6$, a much broader region of *best solvers* and also a higher overall score. In contrast, an initial value of $p_1 = 1$ yields a lower overall score as well as a smaller region of *best solvers*. All simulation results show a similar pattern in response to parameter variations, with the *highest scoring* parameter combinations located in a wedge-shaped area at the center of the plot and lower scores in both lower left and upper right corners. Please refer to *Cognitive Interpretation* further below for a detailed discussion of these results.

Choice of Optimization Methods

In this section we discuss the results of our simulation study regarding the numerical differentiability of our reformulated model as well as the appropriate optimization methods. In order to apply derivative-based optimization methods, a tractable formulation is necessary. However, the ACT-R architecture contains many logical statements (e.g., *if-then-else*) and absolute value expressions that challenge a derivative-based approach. As shown before, such non-differentiabilities can be smoothed using continuous approximations of the Heaviside and Delta functions Eq (9). This is similar to the approach of Gurney et al. [23], in which the authors also used a smooth approximation of the Heaviside function in order to model action selection.

A different approach is described by Chapelle et al. [24], who used a softmax activation function and showed that it is possible to perform gradient-based optimization on their smooth approximation. This approach however requires i.a. that the index selected by argmax is unique. How to deal with chunks having (almost) the same activation remains an open question.

Choice of the Smoothing Parameter *h* We concentrated on the influence of the parameter h of the Heaviside function, as the parameter a of the Delta function turned out to be uncritical. Larger values of h correspond to sharper transitions at $x = 0$. To identify the value of h for which the model becomes numerically differentiable, we ran simulations for $h \in \{0.1, 1.0, 10, 10^2, \ldots, 10^7\}$ with $P = 20$, in the particular parameter interval $\tau \in [-3.16, -3.12]$ sampled at step size $\Delta\tau = 10^{-5}$. We also separately varied h for smoothing of the similarity calculation, denoted by h_{sim}, smoothing of argmax, denoted by h_{argmax}, and for computation of sugar production and workers, denoted by h_{env}.

Results in Fig 6 show that, the *argmax* term proves to be critical for matching the behavior of the *Sugar Factory* model for the ACT-R and the Python implementation. Larger values of h (Fig 6, left) are required but yield a numerically non-differentiable function. Decreasing the values of h_{argmax} leads i.a. to a random choice of chunks that undermines the learning process. Fig 6 shows that the score drops from about 19.5 (left) to approximately 5.2 (center). For the similarity calculation and the calculation of sugar production and workers, the choice of h is

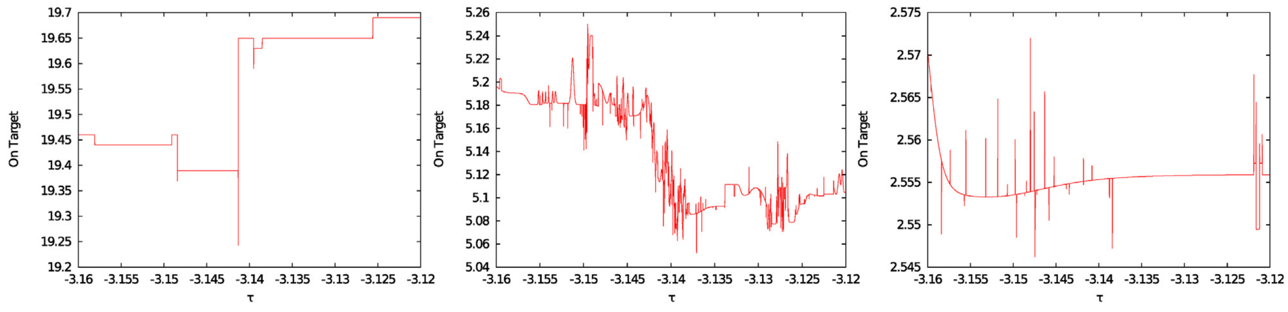

Fig 6. Mean value of rounds on target for 100 inputs over 40 rounds with *P* = const. Left: $h_{\text{argmax}} = 10^7$, $h_{\text{sim}} = h_{\text{env}} = 10$. Center: $h_{\text{argmax}} = 10^3$, $h_{\text{sim}} = h_{\text{env}} = 10$. Right: $h_{env} = 10$, $h_{\text{argmax}} = h_{\text{sim}} = 100$. Note the different vertical axis ranges.

doi:10.1371/journal.pone.0158832.g006

less critical, but $h_{\text{argmax}} = 100$ is still too large to yield a numerically differentiable model (right).

We may conclude that, even though smoothing the argmax can be a feasible approach, cf. [23, 24], precise modeling of the argmax is crucial at least for the particular case of the *Sugar Factory* model.

Heuristic and Mathematical Optimization Approaches Optimization methods such as genetic algorithms [25] or particle swarm [26] search the global parameter space based on *heuristics*. Such algorithms however rely on the computational time, for example, as termination criterion, as they have little information on whether or not they have actually found the an optimum. Two examples for such heuristic optimization methods are ESCH [27], a modified evolutionary algorithm, and Controlled Random Search (CRS) with local mutation [28]. CRS starts with a population of random points, and evolves them heuristically, a method comparable to genetic algorithms.

In contrast, *mathematical optimization solvers* are characterized by the use of derivatives as a source of additional information to make sure that an optimum is reached. Those *mathematical* optimization solvers are e.g. Newton-type algorithms, e.g. [29], or steepest descent methods, e.g. [30], but also include derivative-free methods such as Nelder-Mead [31] or BOBYQA [32], which approximate the behavior of gradient based solvers. Nelder-Mead is a downhill simplex method while BOBYQA uses an iteratively constructed quadratic approximation for the objective function. Whereas *heuristic* optimization methods are quickly stretched to their limits with an increasing dimensionality of the parameter space, the number of iterations for mathematical optimization methods, in particular for derivative based ones, ideally is independent of the problem dimensions.

Numerical Results for the Sugar Factory

We applied a selection of heuristic and mathematical optimization algorithms that promise to cope with the non-differentiability of the nonlinear recurrence model. Our selection comprises Nelder-Mead Simplex [33] with explicit support for bound constraints [34], BOBYQA, ESCH, and CRS. All optimization algorithms were applied using the Python interface NLopt [35].

The stopping criterion for BOBYQA and Nelder-Mead was a relative tolerance on the optimization parameters of 0.1. For the heuristic global solvers ESCH and CRS we successively increased the time limit up to about 1000s. The stopping criterion was then set to the minimum run time for which there was no improvement of the found maxima observed.

Fit optimization Table 1 shows the results for the best fit to human reference performance, with $R_{\text{ref}} = 7.9$ taken from the literature [15]. Using multiple start values, all solvers found approximately the same point as a maximum. For ESCH and CRS the results displayed are for a time limit of 5.0 seconds.

Performance optimization For the single input displayed in Fig 3, all solvers found the global maximal score of 31, depending on suitable choices of the initial values for parameters τ

Table 1. Maxima found by different solvers for $n = 100$ inputs. Objective was to find the parameter combination best fitting a human reference value using RMSD (fit optimization).

Solver	τ	P	Max.	#Eval.
Nelder-Mead	0.5	28.13	4.05	67
BOBYQA	0.5	27.80	4.05	54
ESCH	0.45	27.88	4.05	6,374
CRS	0.48	32.94	4.05	4,500

doi:10.1371/journal.pone.0158832.t001

Table 2. Maxima found by different solvers for $n = 100$ inputs. Objective was to find the parameter combination with the best score (performance optimization).

Solver	τ	*P*	Max.	#Eval.
Nelder-Mead	-4.00	27.00	20.15	36
BOBYQA	-4.00	27.00	20.15	43
ESCH	-3.13	22.36	20.13	863
CRS	-4.21	28.52	20.2 0	860

doi:10.1371/journal.pone.0158832.t002

and *P*. Table 2 shows the results for $a = 1$ and $b = 0$ and 100 inputs using multiple start values (see Fig 5, left). The local solvers Nelder-Mead and BOBYQA both found the same local maximum ($\tau = -4.00$, $P = 27.00$ with objective = 20.15). Table 2 shows the maxima found by the heuristic global solvers after 960 seconds (see Fig 7). For $a = 1$ and $b = -1$, all solvers found the same point as a maximum ($\tau \approx -6.5$, $P \approx 30$ with objective $\approx$13.87), except CRS which found a slightly better point ($\tau \approx -8.15$, $P \approx 34.9$ with objective $\approx$14.04).

Fig 8 shows the optimization trajectories for Nelder-Mead and BOBYQA.

Cognitive Interpretation

Our results show how high-performance simulation and optimization can provide insights beyond just optimizing the fit to aggregate human data. Interestingly, optimizing for highest performance shows that the optimal parameter combination is considerably different from the best fit to typical human performance, particularly with respect to parameter τ (see Fig 7). This raises two questions, namely why the τ value of the model with the best fit to human performance diverges from the optimal model, and how a lower τ value leads to better performance (see Fig 5).

Non-optimal human performance A simple answer to the first question is that people do not behave formally optimal in many decision situations [36] in general, and in the *Sugar Factory* in particular [14, 15]. The structure of the human cognitive system seems to be geared towards robust information processing in typical human environments with incomplete or

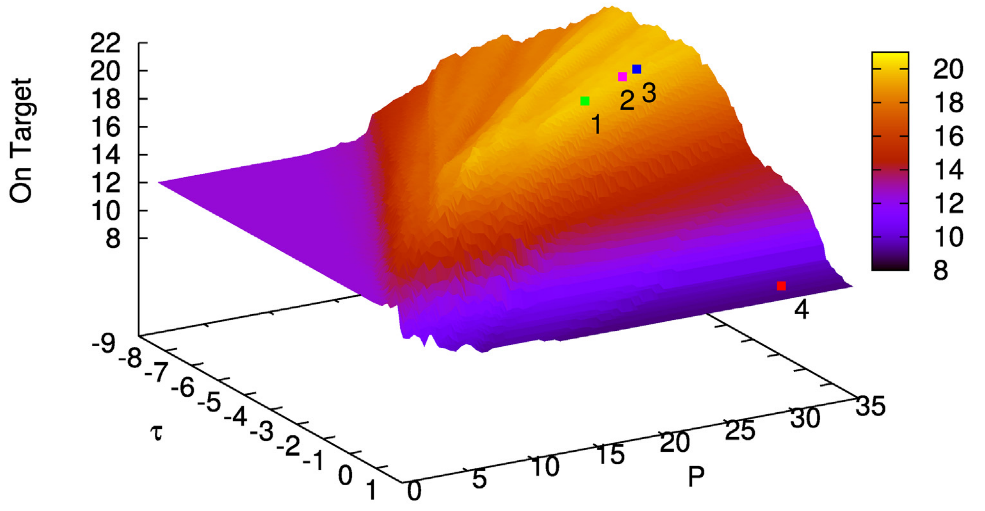

Fig 7. Mean of 100 inputs over 40 rounds, $p_0 = 6$, medium grid (8,256 grid points). Points 1–4 show 1: best score found by ESCH, 2: best score found by local solvers, 3: best score found by CRS, 4: best fit to reference human.

doi:10.1371/journal.pone.0158832.g007

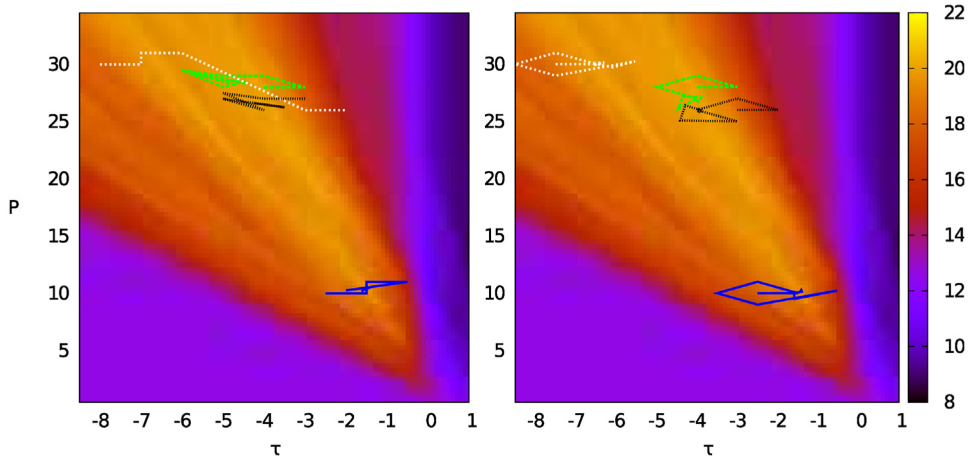

Fig 8. Optimization trajectories for Nelder-Mead (left) and BOBYQA (right) with four different start values.

doi:10.1371/journal.pone.0158832.g008

uncertain information [37, 38], rather than formal optimization given strict assumptions. Another possibility is that the model of human cognition used in this study is not valid to start with. However, given that existing studies of the *Sugar Factory* task and its derivatives show that implicit learning is a strong explanatory mechanism [14, 15], that the implementation of implicit learning based on the ACT-R architecture generally matches human data in other studies [39], and that the specific model used here has been empirically supported [16], we think there are good grounds to assume that the basic structure of the model is appropriate.

Influence of the parameters τ and P The second question is how a lower τ value leads to better performance (see Fig 5). Apparently, being open to considering vague memories (i.e., a low retrieval threshold τ) is mostly a good strategy in this task. However, this may be a task-specific effect, as participants are provided only with noise-free and correct information. Any memory, however vague, that is sufficiently similar to the target situation is therefore on average a useful source of information and likely to increase performance. If the recall threshold τ is high, considering chunks with a lower similarity (low similarity weighting P) produces a related effect. However, in contrast to τ, the best-fit estimate for parameter P lies close to the theoretical optimum for this task. The more conservative memory threshold (a high τ value) shown by most human participants may represent a suitable trade-off across a range of different situations, given that information often is noisy or unreliable and a higher selectivity therefore advisable. This is supported by the fact that the best-fit value of $\tau = 0.5$ we found is close to the value recommended by the architecture as a suitable default for many situations ($\tau = 0$) [40].

Influence of different initial values We also investigated how choosing different initial values for the sugar production task influenced performance. We observed that the effectiveness of instance-based learning in this task noticeably depends on the initial production values. An initial value of $p = 9$ yields the best overall performance, as it is part of the target value range, $8 \leq p \leq 10$, and therefore produces memory chunks of trials *on target* early, which is important for guiding control behavior. This insight is practically relevant for behavioral studies, as the sensitivity to starting conditions has so far not been considered in studies using the *Sugar Factory*.

Discussion

Cognitive architectures are powerful tools for modeling higher-level cognition. However, methods for efficient exploration of parameter spaces and quick parameter estimation in these models are still in the process of development. In this article, we have demonstrated first steps towards a differentiable formulation of an instance-based learning model implemented in the ACT-R cognitive architecture. We conducted a simulation study for an analysis of central model parameters and showed how mathematical optimization techniques can be applied for efficient parameter identification.

We implemented a mathematical reformulation of the *Sugar Factory*, a simple instance-based learning task, in *Python* and showed that it is possible to derive a tractable formulation. The generic part of this formulation, related to the ACT-R declarative memory module, can in principle be transferred to ACT-R models of other tasks like Backgammon [41], Air-Traffic Control [42], the beer game [43], or even more complex tasks like the Water Purification Plant [39].

Furthermore, our approach should be transferable to other ACT-R models relying on the declarative memory module. Currently we are working on a reformulation of a model of the *Digit Span Task*, which includes i.a. associative activation for modeling working memory. The approach could also be extended to cover the procedural module of ACT-R. For example, Gurney et al. (2001) [23] described a mathematical model of procedural action selection that shows several parallels to the reformulation of declarative memory presented here.

We conducted simulation studies to determine model properties by varying the parameter h of an approximation of the Heaviside function, which we used for smoothing the non-differentiable parts of our model. The simulations showed that in order to obtain exactly the same results like the standard ACT-R model a large h for the smoothened *argmax* is necessary, contrary to other parts of our model like the similarity calculation and the environmental setting (i.e. calculation of sugar production and workers). This however, leads to piecewise constant behavior of our *Python* implementation. For smaller h our model becomes numerically differentiable, but at the same time the learning process is replaced by random behavior. Therefore, at this stage, even though derivatives can be calculated, using gradient-based optimization methods is not feasible.

We then showed how to address two common optimization problems: Firstly, the identification of parameter values that result in a best model fit to human reference values, and, secondly, the determination of parameter values that maximize the score of a scenario. We applied both heuristic and mathematical optimization algorithms that promise to cope with the non-differentiability of our nonlinear recurrence model and showed that *mathematical optimization* solvers like Nelder-Mead Simplex or BOBYQA turned out to be the best choice for the model at hand. Not only do they have the advantage of using approximations of the derivatives to determine if an extremum is found, thus needing a lower number of iterations than the *heuristic optimization* solvers, but they are also, in principle, able to deal with higher dimensional problems. Furthermore, we conducted a simulation study for two central model parameters, the retrieval threshold τ and the similarity weighting P, using high-performance computing. Results revealed a sensitivity of the task to initial settings, a result which has not been considered in the experimental uses of the *Sugar Factory* task so far. Our findings also indicate that human performance in this specific tasks seems to be hampered in part by a tendency to be overly conservative in which memory instances are considered.

Outlook

As the *argmax* turned out to be the crucial part of the transcribed ACT-R model, we pursued a non-differentiable approach in this article and developed a nonlinear recurrence relation that could be optimized with a selection of heuristic or derivative-free solvers. This approach has the advantage of allowing for the computation of a single round of the cognitive process by a mere function evaluation.

We envision in a next step to derive exact reformulations of IBL problems and ACT-R cognitive processes that are amenable to derivative-based optimization methods, as follows: Returning once more to the statement $i^* = \operatorname{argmax}\{A_i\}$ for data $A_1, \ldots, A_k$, consider the following constrained optimization problem:

$$\begin{cases} \min\limits_{A^*,w} & A^* & & & \\ s.t. & A^* & \geq & A_i, & 1 \leq i \leq k, \\ w_i \cdot (A_i - A^*) & & \geq & 0, & 1 \leq i \leq k, \\ & w_i & \in & [0,1], \quad \sum\limits_{i=1}^{k} w_i = 1. & \end{cases}$$

Herein, A^* is a free variable set to the maximum activation value by virtue of minimization and the first inequality. We seek the argmax, i.e. the index i^* with $A_{i^*} = A^*$. All differences in the second inequality are non-positive, and all with $A_i < A^*$ are negative. This forces the corresponding indicators w_i to zero. Then, the indicator w_{i^*} is forced to one by the equality in the third line. A function $f(i^*)$ depending on i^*, the argmax, may then be expressed as

$$f(i^*) = \sum_{i=1}^{k} w_i f(i),$$

which is now bi-linear, differentiable, and independent of the argmax, but yields the same value because $w_i = 0$ for $i \neq i^*$, and $w_{i^*} = 1$.

This formulation represents the computation of one sample of the dynamic decision making process by the solution of a bi-linear optimization problem. The approach is hence significantly more demanding in terms of computational effort. Moreover, optimizing over process samples computed in this way constitutes a bi-level optimization problem. Treatment of such problems is significantly more demanding also in terms of mathematical optimization algorithms, but has the advantage of precisely reproducing the sequence of chunk activations as determined by ACT-R.

Another possibility that might increase the tractability of our model is a different representation of the production rules, as in [44]. Instead of using a two-step approach like in ACT-R, production rules only have one feature, their utilities.

Acknowledgments

The authors gratefully acknowledge support by the Excellence Initiative of the German Research Council (DFG) and by DFG Graduate School 220 (Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences).

Author Contributions

Conceived and designed the experiments: NS ME CK SK DVH. Performed the experiments: NS ME. Analyzed the data: NS ME. Wrote the paper: NS ME CK SK DVH.

References

1. Anderson JR, Bothell D, Byrne MD, Douglass S, Lebiere C, Qin Y. An integrated theory of the mind. Psychological review. 2004; 111(4):1036–1060. doi: 10.1037/0033-295X.111.4.1036 PMID: 15482072
2. Laird JE, Newell A, Rosenbloom PS. Soar: An architecture for general intelligence. Artificial intelligence. 1987; 33(1):1–64. doi: 10.1016/0004-3702(87)90050-6

3. Laird JE. Extending the Soar cognitive architecture. Frontiers in Artificial Intelligence and Applications. 2008; 171:224–236.
4. O'Reilly RC, Hazy TE, Herd SA. The leabra cognitive architecture: how to play 20 principles with nature and win! Oxford University Press; 2012.
5. Eliasmith C. How to build a brain: A neural architecture for biological cognition. Oxford University Press; 2013.
6. Lewandowsky S, Farrell S. Computational modeling in cognition: Principles and practice. Sage; 2010.
7. Dawson MRW. Minds and machines: Connectionism and psychological modeling. John Wiley & Sons; 2008.
8. Roberts S, Pashler H. How persuasive is a good fit? A comment on theory testing. Psychological review. 2000; 107(2):358–367. doi: 10.1037/0033-295X.107.2.358 PMID: 10789200
9. Best BJ, Furjanic C, Gerhart N, Fincham J, Gluck KA, Gunzelmann G, et al. Adaptive mesh refinement for efficient exploration of cognitive architectures and cognitive models. Proc Ninth Int Conf Cognitive Modeling. 2009;p. 1–6.
10. Gluck K, Scheutz M, Gunzelmann G, Harris J, Kershner J. Combinatorics meets processing power: Large-scale computational resources for BRIMS. In: Proc. Sixteenth Conference on Behavior Representation in Modeling and Simulation; 2007. p. 73–83.
11. Kase SE, Ritter FE, Schoelles M. From modeler-free individual data fitting to 3-D parametric prediction landscapes: A research expedition. In: Proc. 30th annual conference of the Cognitive Science Society; 2008. p. 1398–1403.
12. Lane PCR, Gobet F. Evolving Non-Dominated Parameter Sets. Journal of Artificial General Intelligence. 2013; 4(1):358–367. doi: 10.2478/jagi-2013-0001
13. Moore LRJ. Cognitive model exploration and optimization: a new challenge for computational science. Computational and Mathematical Organization Theory. 2011; 17(3):296–313. doi: 10.1007/s10588-011-9092-8
14. Berry DC, Broadbent DE. On the relationship between task performance and associated verbalizable knowledge. The Quarterly Journal of Experimental Psychology. 1984; 36(2):209–231. doi: 10.1080/14640748408402156
15. Dienes Z, Fahey R. Role of specific instances in controlling a dynamic system. Journal of experimental psychology: learning, memory, and cognition. 1995; 21(4):848–862.
16. Taatgen NA, Wallach D. Whether skill acquisition is rule or instance based is determined by the structure of the task. Cognitive Science Quarterly. 2002; 2(2):163–204.
17. Anderson JR. How can the human mind occur in the physical universe? New York, NY, US: Oxford University Press; 2007.
18. Anderson JR. Human symbol manipulation within an integrated cognitive architecture. Cognitive science. 2005; 29(3):313–341. doi: 10.1207/s15516709cog0000_22 PMID: 21702777
19. Marcus GF. The Algebraic Mind: Integrating Connectionism and Cognitive Science. MIT Press; 2003.
20. Chung PH, Byrne MD. Cue effectiveness in mitigating postcompletion errors in a routine procedural task. International Journal of Human-Computer Studies. 2008; 66(4):217–232. doi: 10.1016/j.ijhcs.2007.09.001
21. Logan GD. Toward an instance theory of automatization. Psychological review. 1988; 95(4):492. doi: 10.1037/0033-295X.95.4.492
22. Gonzalez C, Lebiere C. Instance-based cognitive models of decision-making. In: Zizzo DJ, Courakis A, editors. Transfer of knowledge in economic decision making. Palgrave McMillan; 2005.
23. Gurney K, Prescott TJ, Redgrave P. A computational model of action selection in the basal ganglia. II. Analysis and simulation of behaviour. Biological cybernetics. 2001; 84(6):411–423. doi: 10.1007/PL00007984 PMID: 11417053
24. Chapelle O, Wu M. Gradient descent optimization of smoothed information retrieval metrics. Information Retrieval. 2010; 13(3):216–235. doi: 10.1007/s10791-009-9110-3
25. Mitchell M. An introduction to genetic algorithms. MIT press; 1998.
26. Trelea IC. The particle swarm optimization algorithm: convergence analysis and parameter selection. Information processing letters. 2003; 85(6):317–325. doi: 10.1016/S0020-0190(02)00447-7
27. Beyer HG, Schwefel HP. Evolution strategies-A comprehensive introduction. Natural computing. 2002; 1(1):3–52. doi: 10.1023/A:1015059928466
28. Kaelo P, Ali MM. Some variants of the controlled random search algorithm for global optimization. JOTA. 2006; 130(2):253–264. doi: 10.1007/s10957-006-9101-0

29. Gill PE, Murray W. Newtontype methods for unconstrained and linearly constrained optimization. Mathematical Programming. 1974; 7(1):311–350. doi: 10.1007/BF01585529

30. Sun W, Yuan YX. Optimization Theory and Methods: Nonlinear Programming. Springer Science & Business Media; 2006.

31. Lagarias JC, Reeds JA, Wright MH, Wright PE. Convergence properties of the Nelder-Mead simplex method in low dimensions. SIAM Journal on optimization; 9(1):112–147. doi: 10.1137/S1052623496303470

32. Powell MJD. The BOBYQA algorithm for bound constrained optimization without derivatives; 2009. NA2009/06.

33. Nelder JA, Mead R. A simplex method for function minimization. The computer journal. 1965; 7(4):308–313. doi: 10.1093/comjnl/7.4.308

34. Box MJ. A new method of constrained optimization and a comparison with other methods. The Computer Journal. 1965; 8(1):42–52. doi: 10.1093/comjnl/8.1.42

35. Johnson SG. The NLopt nonlinear-optimization package; 2010. http://ab-initio.mit.edu/nlop.

36. Klein G. The fiction of optimization. Bounded rationality: The adaptive toolbox. 2002; 103:114.

37. Gigerenzer G, Gaissmaier W. Heuristic decision making. Annual review of psychology. 2011; 62:451–482. doi: 10.1146/annurev-psych-120709-145346 PMID: 21126183

38. Walsh MM, Gluck KA. Mechanisms for Robust Cognition. Cognitive Science. 2015; 39(6):1131–1171. doi: 10.1111/cogs.12192 PMID: 25352094

39. Gonzalez C, Lerch JF, Lebiere C. Instance-based learning in dynamic decision making. Cognitive Science. 2003; 27(4):591–635. doi: 10.1207/s15516709cog2704_2

40. ACT-R 6 0 Tutorial; 2012. Retrieved from the ACT-R Web site:http://act-r.psy.cmu.edu/software/.

41. Sanner S, Anderson JR, Lebiere C, Lovett MC. Achieving efficient and cognitively plausible learning in backgammon. Proc Seventeenth Int Conf Machine Learning. 2000;p. 823–830.

42. Lebiere C, Anderson JR, Bothell D. Multi-tasking and cognitive workload in an ACT-R model of a simplified air traffic control task. Proceedings of the Tenth Conference on Computer Generated Forces and Behavior Representation. 2001.

43. Martin MK, Gonzalez C, Lebiere C. Learning to make decisions in dynamic environments: ACT-R plays the beer game. Proceedings of the Sixth International Conference on Cognitive Modeling. 2004.

44. Stewart TC, Eliasmith C. Building production systems with realistic spiking neurons. In: Proceedings of the 30th annual meeting of the Cognitive Science Society. Austin, TX: Cognitive Science Society; 2008.