Hierarchical Reinforcement Learning Explains Task Interleaving Behavior

Christoph Gebhardt¹ · Antti Oulasvirta² · Otmar Hilliges¹

Accepted: 21 September 2020 / Published online: 5 November 2020
© The Author(s) 2020

Abstract

How do people decide how long to continue in a task, when to switch, and to which other task? It is known that task interleaving adapts situationally, showing sensitivity to changes in expected rewards, costs, and task boundaries. However, the mechanisms that underpin the decision to stay in a task versus switch away are not thoroughly understood. Previous work has explained task interleaving by greedy heuristics and a policy that maximizes the marginal rate of return. However, it is unclear how such a strategy would allow for adaptation to environments that offer multiple tasks with complex switch costs and delayed rewards. Here, we develop a hierarchical model of supervisory control driven by reinforcement learning (RL). The core assumption is that the supervisory level learns to switch using task-specific approximate utility estimates, which are computed on the lower level. We show that a hierarchically optimal value function decomposition can be learned from experience, even in conditions with multiple tasks and arbitrary and uncertain reward and cost structures. The model also reproduces well-known key phenomena of task interleaving, such as the sensitivity to costs of resumption and immediate as well as delayed in-task rewards. In a demanding task interleaving study with 211 human participants and realistic tasks (reading, mathematics, question-answering, recognition), the model yielded better predictions of individual-level data than a flat (non-hierarchical) RL model and an omniscient-myopic baseline. Corroborating emerging evidence from cognitive neuroscience, our results suggest hierarchical RL as a plausible model of supervisory control in task interleaving.

Keywords Computational modeling · Task interleaving · Hierarchical reinforcement learning · Bayesian inference · Hierarchical reinforcement learning model for task interleaving

Introduction

How long will you keep reading this paper before you return to email? Knowing when to persist and when to do something else is a hallmark of cognitive functioning and is intensely studied in the cognitive sciences (Altmann and Trafton 2002; Brumby et al. 2009; Duggan et al. 2013; Janssen and Brumby 2010; Jersild 1927; Monsell 2003; Norman and Shallice 1986; Oberauer and Lewandowsky 2011; Payne et al. 2007; Wickens and McCarley 2008). In the corresponding decision problem, *the task interleaving*

problem, an agent must decide how to share its resources among a set of tasks over some period of time. We here investigate *sequential* task interleaving, where only one demanding task is processed at a time. This problem is set apart from the concurrent multitasking problem, which involves simultaneous resource-sharing (Brumby et al. 2018; Oberauer and Lewandowsky 2011; Salvucci and Taatgen 2008).

The task interleaving problem is a decision-making problem: The agent can focus on a task, thus advancing it and collecting its associated rewards. It can also switch to another task, but this incurs a switch cost, the magnitude of which depends on the agent's current state (Jersild 1927; Monsell 2003). Consider the two-task interleaving problem shown in Fig. 1: How would you interleave and how would a rational agent behave? The general problem is non-trivial: our everyday contexts offer large numbers of tasks with complex and uncertain properties. Yet, most of the time people interleave tasks with apparent ease.

✉ Christoph Gebhardt
cgebhard@ethz.ch

¹ Eidgenössische Technische Hochschule Zurich,
Stampfenbachstrasse 48, 8092 Zürich, Switzerland

² Aalto University, Konemiehentie 2, 02150 Espoo, Finland

Fig. 1 Example of the task interleaving problem with two tasks: Given a limited time window and N tasks with reward/cost structures, an agent has to decide what to focus on at any given time such that the

totally attained reward gets maximized. Attending a task progresses its state and collects the associated rewards $r_T(s)$, while switching to another task incurs a cost $c_T(s)$

It is well-known that human interleaving behavior is adaptive. In particular, the timing of switches shows sensitivity to task engagement (Janssen and Brumby 2015; Wickens and McCarley 2008). Factors that define the engagement of a task are interest (Horrey and Wickens 2006) and priority (Iani and Wickens 2007), commonly modeled as in-task rewards. Task interleaving is also sensitive to interruption costs (Trafton et al. 2003) and to resumption costs (Altmann and Trafton 2002; Gutzwiller et al. 2019; Iqbal and Bailey 2008). These costs represent additional processing demands due to the need to alternate back and forth between different tasks and the resulting additional time it takes to complete them (Jersild 1927; Oberauer and Lewandowsky 2011). This is affected by skill-level (Janssen and Brumby 2015) and memory recall demands (Altmann and Trafton 2007; Oulasvirta and Saariluoma 2006). In addition, task switches tend to be pushed to boundaries between tasks and subtasks because a task can be resumed more rapidly on return when it was left at a good stopping point as switch costs are lower (Altmann and Trafton 2002; Janssen et al. 2012; McFarlane 2002).

Previous models have shed light on possible mechanisms underlying these effects: (i) According to a time-based switching heuristic, the least attended task receives resources, to balance resource-sharing among tasks (Salvucci and Taatgen 2008; Salvucci et al. 2009), or in order to refresh it in memory (Oberauer and Lewandowsky 2011); (ii) According to a foraging-based model, switching maximizes in-task reward (Payne et al. 2007; Duggan et al. 2013), which is tractable for diminishing-returns reward functions using the marginal value theorem; (iii) According to a multi-attribute decision model, task switches are determined based on task attractiveness, defined by importance, interest, and difficulty (Wickens et al. 2015).

While these models have enhanced our understanding, we still have an incomplete picture of how human inter-

leaving adapts to multiple tasks and complex reward/cost structures, including delayed rewards. Examples with non-diminishing rewards are easy to construct: in food preparation, the reward is collected only after cooking has finished. In choosing cooking over Netflix, people demonstrate an ability to avoid being dominated by immediately achievable rewards. In addition, we also need to explain people's ability to interleave tasks they have not experienced before. If you have never read this paper, how can you decide to switch away to email or continue to read?

Here we propose *hierarchical reinforcement learning* (HRL) as a unified account of adaptive supervisory control in task interleaving. While there is extensive work on HRL in machine learning, we propose it here specifically as a model of *human* supervisory control that keeps track of ongoing tasks and decides which to switch to (Norman and Shallice 1986; Wickens and McCarley 2008). We assume a two-level supervisory control system, where both levels use RL to approximate utility based on experience.

From a machine learning perspective, RL is a method for utility approximation in conditions that are uncertain and where gratifications are delayed (Sutton and Barto 1998). In task interleaving, we use it to model how people estimate the value of continuing in a task and can anticipate a high future reward even if the immediate reward is low. *Hierarchical* RL extends this by employing temporal abstractions that describe state transitions of variable durations. Hierarchicality has cognitive appeal thanks to its computational tractability. Selecting among higher level actions reduces the number of decisions required to solve a problem (Botvinick 2012). We demonstrate significant decreases in computational demands when compared with a flat agent equal in performance.

Emerging evidence has shed light on the neural implementation of RL and HRL in the human brain. The temporal difference error of RL correlates with dopamine

signals that update reward expectation in the striatum and also explains the release of dopamine related to levels of uncertainty in neurobiological systems (Gershman and Uchida 2019). The prefrontal cortex (PFC) is proposed to be organized hierarchically for supervisory control (Botvinick 2012; Frank and Badre 2011) such that dopaminergic signaling contributes to temporal difference learning and PFC representing currently active subroutines. As a consequence, HRL has been applied to explain brain activity during complex tasks (Botvinick et al. 2009; Rasmussen et al. 2017; Balaguer et al. 2016). However, no related work considers hierarchically optimal problem decomposition of cognitive processes in task interleaving. Hierarchical optimality is crucial in the case of task interleaving, since rewards of the alternative tasks influence the decision to continue the attended task.

To test the idea of hierarchical RL, it is necessary to develop computational models that are capable of performing realistic tasks and replicating human data closely by reference to neurobiologically plausible implementation (Kriegeskorte and Douglas 2018). Computational models that generate task performance can expose interactions among cognitive components and thereby subject theories to critical testing against human behavior. If successful, such computational models can, in turn, serve as reference and inspiration for further research on neuroscientific events and artificial intelligence methods. However, such models will unnecessarily involve substantial parametric complexity, which calls for methods from Bayesian inference and large behavioral datasets (Kangasrääsiö et al. 2019).

In this spirit, we present a novel computational implementation of HRL for task interleaving and assess it against a rich set of empirical findings. The defining feature of our implementation is a two-level hierarchical decomposition of the RL problem. (i) On the lower—or task type—level, a state-action value function is kept for each *task type* (e.g., writing, browsing) and updated with experience of each ongoing *task instance* (e.g., writing task A, browsing task B, browsing task C). (ii) On the higher—or task instance—level, a reference is kept to each ongoing task instance. HRL decides the next task based on value estimates provided from the lower level. This type-instance distinction permits taking decisions without previously experiencing the particular task instance. By modeling task type-level decisions with a *semi-Markov decision process* (SMDP), we model how people decide to switch at *decision points* rather than at a fixed sampling interval. In addition, the HRL model allows learning arbitrarily shaped reward and cost functions. For each task, a reward and a cost function is defined over its states (see Fig. 1).

While the optimal policy of hierarchically optimal HRL and flat RL produce the same decision sequence given the same task interleaving problem, differences in how this policy is learned from experience render HRL a cognitively more plausible model than flat RL. Flat RL learns expected rewards for each task type to task type transition and, hence, needs to observe a particular transition in training to be able to make a rational decision at test time. In contrast, through hierarchical decomposition, our HRL agent does not consider the task from which a switch originates, but learns expected rewards for transitions by only considering switch destination. This enables the HRL agent to make rational decisions for task type to task type switches that have not been observed in training when task types themselves are familiar. We hypothesize that this better matches with human learning of task interleaving.

Modeling task interleaving with RL assumes that humans learn by trial and error when to switch between tasks to maximize the attained reward while minimizing switching costs. For the example in Fig. 1, this means that they have learned by conducting several writing tasks that the majority of its reward is attained at its end (states 8 and 9). In addition, they experienced that a natural break point is reached when finishing a paragraph (states 6 and 7) and that one can switch after that without any induced costs. Similarly, they learned that switching to and from a browsing task is generally not very costly due to its simplicity. However, also its reward quickly diminishes as no interesting new information can be attained. The acquired experiences are encoded in memory, which provides humans with an intuition on which task to attend in unseen similar situations. RL also considers the human capability to make decisions based on future rewards that cannot be attained immediately. For instance, it can depict the behavior of a human that finishes the writing task to attain the high reward at its end while inhibiting to switch to the browsing task that would provide a low immediate gratification.

In the rest of the paper, we briefly review the formalism of hierarchical reinforcement learning before presenting our model and its implementation. We then report evidence from simulations and empirical data. The model reproduces known patterns of adaptive interleaving and predicts individual-level behavior measured in a challenging and realistic interleaving study with six tasks ($N = 211$). The HRL model was better or equal than RL- and omniscient-myopic baseline models, which does not consider long-term rewards. HRL also showed more human-like patterns, such as sensitivity to subtask boundaries and delayed gratification. We conclude that human interleaving behavior

appears better described by hierarchically decomposed optimal planning under uncertainty than by heuristics, or myopic, or flat RL strategies.

Background—Hierarchical Reinforcement Learning

Markov and Semi-Markov Decision Processes

The family of Markov decision processes (MDP) is a mathematical framework for decision-making in stochastic domains (Kaelbling et al. 1998). The MDP is a four-tuple (S, A, P, R) , where S is a set of states, A a set of actions, P the state transition probability for going from a state s to state s' after performing action a (i.e., $P(s'|s, a)$), and R the reward for action a in state s (i.e., $R : S \times A \rightarrow \mathbb{R}$). The expected discounted reward for action a in s when following policy π is known as the Q value: $Q^\pi(s, a) = E_{s_t}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$, where γ is a discount factor. Q values are related via the Bellman equation: $Q^\pi(s, a) = \sum_{s'} P(s'|s, a)[R(s', s, a) + \gamma Q^\pi(s', \pi(s'))]$. The optimal policy can then be computed as $\pi^* = \arg \max_a Q^\pi(s, a)$. Classic MDPs assume a uniform discrete step size. To model temporally extended actions, semi-Markov decision processes (SMDPs) are used. SMDPs represent snapshots of a system at decision points where the time between transitions can be of variable temporal length. An SMDP is a five-tuple (S, A, P, R, F) , where S, A, P , and R describe an MDP and F gives the probability of transition times for each state-action pair. Its Bellman equation is:

$$Q^\pi(s, a) = \sum_{s', t} F(t|s, a) P(s'|s, a) [R(s, a) + \gamma^t Q^\pi(s', \pi(s'))], \quad (1)$$

where t is the number of time units after the agent chooses action a in state s and $F(t|s, a)$ is the probability that the next decision epoch occurs within t time units.

Reinforcement Learning

Reinforcement learning solves Markov decision processes by learning a state-action value function $Q(s, a)$ that approximates the Q value of the Bellman equation $Q^\pi(s, a)$. There are two classes of algorithms for RL: model-based and model-free algorithms. In model-based algorithms, the state transition probabilities $F(t|s, a)$ and $P(s'|s, a)$ are known and policies are found by enumerating the possible sequences of states that are expected to follow

a starting state and action while summing the expected rewards along these sequences. In this paper, we use model-free RL algorithms to solve an MDP. These algorithms learn the approximate state-action value function $Q(s, a)$ in an environment where the state transition probability functions $F(t|s, a)$ and $P(s'|s, a)$ are unknown but can be sampled from it. One model-free algorithm that learns the approximate state-action value function via temporal difference learning is *Q-learning*: $Q(s_t, a_t) = Q(s_t, a_t) + \alpha [R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$, where s_t, s_{t+1}, a_t and R_{t+1} are sampled from the environment.

Hierarchical Reinforcement Learning

Hierarchical RL (HRL) is based on the observation that a variable can be irrelevant to the optimal decision in a state even if it affects the value of that state (Dietterich 1998). The goal is to decompose a decision problem into subroutines, encapsulating the internal decisions such that they are independent of all external variables other than those passed as arguments to the subroutine. There are two types of optimality of policies learned by HRL algorithms. A policy which is optimal with respect to the non-decomposed problem is called *hierarchically optimal* (Andre and Russell 2002; Ghavamzadeh and Mahadevan 2002). A policy optimized within its subroutine, ignoring the calling context, is called *recursively optimal* (Dietterich 1998).

Computational Model of Task Interleaving

Task Model

We model tasks via the reward $r_T(s)$ and cost $c_T(s)$ functions defined over discrete states s (see Fig. 2). The reward represents subjective attractiveness of a state in a task (Norman and Shallice 1986; Wickens and McCarley 2008). The cost represents overheads caused by a switch to a task (Jersild 1927; Oberauer and Lewandowsky 2011; Oulasvirta and Saariluoma 2006). A state is a discrete representation of progress within a task and the progress is specific to a task type. For instance, in our reading task model, progress is approximated by the position of the scroll bar in a text box. Reward and cost functions can be arbitrarily shaped. This affords flexibility to model tasks with high interest (Horrey and Wickens 2006; Iani and Wickens 2007), tasks with substructures (Bailey and Konstan 2006; Monk et al. 2004), as well as complex interruption and resumption costs (Trafton et al. 2003; Rubinstein et al. 2001).

Fig. 2 An exemplary task model for paper writing. The model specifies in-task rewards with function r_T and resumption/interruption costs with function c_T . Both are specified over a discrete state s that defines progress in a task

Hierarchical Decomposition of Task Environments

Literature sees task interleaving to be decided by a human supervisory control mechanism that keeps track of ongoing tasks and decides which to switch to (Norman and Shallice 1986; Wickens and McCarley 2008). We propose to model this mechanism with hierarchical reinforcement learning, assuming a two-level supervisory control system. Intuitively, this means that we assume humans to make two separate decisions at each task switch: first, when to leave the current task and second, which task to attend next. These decisions are learned with two separate memory structures (i.e., state-action value functions) and updated with experience. The lower level learns to decide whether to continue or leave the current task. Thus, it keeps a state-action value function for each *task type* (e.g., writing,

browsing) and updates it with experience of each ongoing *task instance* (e.g., writing task A, browsing task B, browsing task C). The higher level learns to decide which task to attend next based on the learned reward expectations of the lower level. In contrast, in flat RL, task interleaving is learned with one memory structure and every task switch is a single decision to attend a next task. We explain the difference between flat and hierarchical RL in more detail in “[Comparison with Flat RL](#).”

Figure 3 shows the hierarchical decomposition of the problem. We decompose the task interleaving decision problem into several partial programs that represent the different tasks. Each task is modeled as a behavioral subroutine that makes decision independent from all other tasks only considering the variables passed to it as arguments (see “[Hierarchical Reinforcement Learning](#)” for background). Rectangles represent composite actions that can be performed to call a subroutine or a primitive action. Each subroutine (triangle) is a separate SMDP. Primitive actions (ovals) are the only actions that directly interact with the task environment. The problem is decomposed by defining a subroutine for each task type: $TaskType_1(s)$ to $TaskType_N(s)$. A subroutine estimates the expected cumulative reward of pursuing a task from a starting state s , until the state it expectedly leaves the task. At a given state s , it can choose from the actions of either continuing $Continue(s)$ or leaving $Leaving(s)$ the task. These actions then call the respective action primitives: *continue*, *leave*. The higher level routine *Root*, selects among all available task instances, $Task_{11}(s)$ to $Task_{NN}(s)$, the one which returns the highest expected reward. When a task instance is selected, it calls its respective task type subroutine passing its in-task state s (e.g., $Task_{11}(s)$ calls $TaskType_1(s)$).

Reward Functions

We define two reward functions. On the task type level, the reward for proceeding with a subroutine from its current

Fig. 3 A hierarchical decomposition of the task interleaving problem: subroutines are triangles, rectangles are composite actions and primitive actions are ovals. *Root* chooses among all available task

instances, e.g., $Task_{11}(s)$, which in turn call the subroutine of their respective type, e.g., $TaskType_1(s)$. A subroutine can either continue $Continue(s)$ or leave $Leave(s)$ a task

state s with action a is:

$$R_t(s, a) = \begin{cases} -c_T(s) & \text{if } a \text{ is } leave \\ r_T(s) & \text{if } a \text{ is } continue, \end{cases} \quad (2)$$

where $c_T(s)$ and $r_T(s)$ are the respective cost and reward functions of the task. This covers cases in which the agent gains a reward by pursuing a task ($r_T(s), a = continue$). It also captures human sensitivity to interruption costs (Trafton et al. 2003) and future resumption costs (Altmann and Trafton 2002; McFarlane 2002), when deciding to terminate task execution ($-c_T(s), a = leave$). Finally, it models the effect of decreasing reward as well as increasing effort both increasing the probability of leaving a task (Gutzwiller et al. 2019). On the task instance level, we penalize state changes to model reluctance to continue tasks that require excessive effort to recall relevant knowledge (Altmann and Trafton 2007; Oulasvirta and Saariluoma 2006). The respective reward function is $R_r(s) = -c_T(z(s))$, where s is the state on the root level, $z(s)$ maps s to the state of its child's SMDP, and $c_T(s)$ is again the cost function of the task.

Hierarchical Optimality

Modeling task interleaving with hierarchical reinforcement learning (HRL) raises the question if policies of this problem should be recursively (Dietterich 1998) or hierarchically optimal (Andre and Russell 2002; Ghavamzadeh and Mahadevan 2002). In our setting, recursive optimality would mean that humans decide to continue or leave the currently pursued task only by considering its rewards and costs. However, rewards of the alternative tasks influence a human's decision to continue the attended task. This is captured with *hierarchically optimal* HRL, which can be implemented using the three-part value function decomposition proposed in Andre and Russell (2002): $Q^\pi(s, a) = Q_r^\pi(s, a) + Q_c^\pi(s, a) + Q_e^\pi(s, a)$ where $Q_r^\pi(s, a)$ expresses the expected discounted reward for executing the current action, $Q_c^\pi(s, a)$ completing the rest of a subroutine, and $Q_e^\pi(s, a)$ for all the reward external to this subroutine. Applied on the lower level of our task interleaving hierarchy, it changes the Bellman equation of task type subroutines as follows:

$$\begin{aligned} Q_{type,r}^\pi(s, a) &= \sum_{s'} P_{type}(s'|s, a) R_{type}(s, a) \\ Q_{type,c}^\pi(s, a) &= \sum_{SS(s',t)} F_{type}(t|s, a) P_{type}(s'|s, a) \gamma_{type}^t \\ &\quad Q_{type}^\pi(s', \pi_{type}(s')) \end{aligned}$$

$$\begin{aligned} Q_{type,e}^\pi(s, a) &= \sum_{EX(s',t)} F_{type}(t|s, a) P_{type}(s'|s, a) \gamma_{type}^t \\ &\quad Q_{root}^\pi(p(s'), \pi_{root}(p(s'))) \end{aligned} \quad (3)$$

where $s, a, s', P_{type}, F_{type}, \pi_{type}$, and γ_{type} are the respective functions or parameters of a task type-level semi-Markov decision process (SMDP). π_{root} is the optimal policy on root level, $p(s)$ maps from a state s to the corresponding state in its parent's SMDP, and Q_{root}^π is the Bellman equation on the root level of our HRL model. $SS(s', t)$ and $EX(s', t)$ are functions that return the subset of next states s' and transition times t that are states respectively exit states defined by the environment of the subroutine. The three Q-functions in Eq. 3 specify the respective parts of the three-part value function decomposition of Andre and Russell (2002). The full Bellman equation of a task type subroutine is then defined as

$$Q_{type}^\pi(s, a) = Q_{type,r}^\pi(s, a) + Q_{type,c}^\pi(s, a) + Q_{type,e}^\pi(s, a). \quad (4)$$

On root level, the decomposed Bellman equation is specified as

$$\begin{aligned} Q_{root,r}^\pi(s, a) &= \sum_{s',t} F_{root}(t|s, a) P_{root}(s'|s, a) [(R_{root}(s) \\ &\quad + Q_{type}^\pi(z(s), \pi_{type}(z(s))))] \\ Q_{root,c}^\pi(s, a) &= \sum_{s',t} F_{root}(t|s, a) P_{root}(s'|s, a) \\ &\quad \gamma_{root}^t Q_{root}^\pi(s', \pi_{root}(s')) \end{aligned} \quad (5)$$

where $s, a, P_{root}, F_{root}, \pi_{root}$, and γ_{root} are the respective functions or parameters of the root-level SMDP. $z(s)$ is the mapping function from root level state to the state of its child's SMDP. Again $Q_{root,r}^\pi$ and $Q_{root,c}^\pi$ are the respective parts of the three-part value function decomposition. Note that there is no Q-function to specify the expected external reward as root is not called by another routine. Following Andre and Russell (2002), $Q_{root,r}^\pi(s, a)$ is rewarded according to the expected reward values of its subroutine $Q_{type}^\pi(z(s), \pi_{type}(z(s)))$. In addition, to model reluctance to continue tasks that require excess effort to recall relevant knowledge, it is penalized according to $R_{root}(s)$. The full Bellman equation of the root routine is defined as

$$Q_{root}^\pi(s, a) = Q_{root,r}^\pi(s, a) + Q_{root,c}^\pi(s, a). \quad (6)$$

Decision Processes

We design the decision processes of our HRL agent to model human supervisory control in task interleaving

Fig. 4 Transition graph of the two SMDPs of our HRL model. On root level, S is the supervisory control state and t_i are actions representing available tasks. Once a task is selected its task type-level SMDP

is called. s_i are discrete states representing progress, c_i is the continue action, l_i is the leave action, and e the exit state handing control to root level

(Norman and Shallice 1986; Wickens and McCarley 2008). Our hypothesis is that humans do not learn expected resumption costs for task type to task type transitions. Instead, they learn the resumption costs of each task type separately and compute the expected costs of a switch by adding the respective terms of the two tasks. In parts, this behavior is modeled through the hierarchical decomposition of the task environment, allowing us to learn the cost expectations of leaving the current task and continuing the next task on separate levels. However, it is also necessary to model the SMDPs of the HRL agent accordingly. Figure 4 shows the transition graph of our model. We define a single supervisory state S for the higher level decision process. In this state, our agent chooses among the available tasks by selecting the respective action t_i . This calls the lower level SMDP, where the agent can decide to continue c_i a task and proceed to its next state s_i , or leave it l_i and proceed to the exit state e . Once the exit state is reached, control is handed to the root level and the agent is again situated in S . To avoid reward interactions between state-action pairs on the higher level, we set γ_{root} to zero. While the higher level resembles a multi-armed bandit, HRL allows modeling task interleaving in a coherent and cognitively plausible model.

Modeling State Transition Times

In our HRL model, we assume that primitive actions of the lower, task type level of the hierarchy follow an SMDP rather than an MDP. This models human behavior, as we do not make decisions at a fixed sampling rate, but rather decide at certain *decision points* whether to continue or leave the attended task. To model non-continuous decision rates, an SMDP accounts for actions with varying temporal length by retrieving state transition times from a probability function $F(t|s, a)$ (see Eq. 1). Transition times are used to discount the reward of actions relative to their temporal length.

To be able to solve the task type-level SMDP with model-free RL, our environment also needs to account

for actions with varying temporal length. This is done by sampling a transition time t uniformly at random for each taken action from an unconditioned probability distribution $E_{TP}(t)$, defined for each task type T and participant P . These distributions are computed per participant by saving the transition time of all logged state transitions of a task type in all trials (excluding the test trial). Thus, we log participants' actions every 100 ms. This rate is high enough to ensure that their task switches are not missed and, hence, the correct transition times are used in the model (cf. the shortest time (700 ms) on high workload tasks (Raby and Wickens 1994)).

Simulations

We report simulation results showing how the model adapts to changing cost/reward structures. To this end, the two-task interleaving problem of Fig. 1 is considered. The writing task T_w awards a high reward when completed. Switching away is costly, except upon completing a chapter. The browsing task T_b , by contrast, offers a constant small reward and switch costs are low. In the simulations, we trained the agent for 250 episodes¹, which was sufficient for saturation of expected reward. The HRL agent was trained using the discounted reward HO-MAXQ algorithm (Andre and Russell 2002) (see Appendix 1 for details). In the simulations, the HRL agent was forced to start with the writing task.

Cost and Task Boundaries: In Fig. 5c, the agent only switches to browsing *after* reaching a subtask boundary in writing, accurately modeling sensitivity to costs of resumption (Altmann and Trafton 2002; Gutzwiller et al. 2019; Iqbal and Bailey 2008).

¹We consider an episode finished when all tasks in the task environment are completed.

Fig. 5 Interleaving sequences (a–d) generated by our hierarchical reinforcement learner on the task interleaving problem specified in Fig. 1 for different values of the discount factor γ_{type} . Discount factors specify the length of the RL reward horizon

Reward Structure: The HRL agent is sensitive to rewards (Horrey and Wickens 2006; Iani and Wickens 2007; Norman and Shallice 1986; Wickens and McCarley 2008), as shown by comparison of interleaving trajectories produced with different values of γ_{type} in Fig. 5. For example, when $\gamma_{type} = 0$, only immediate rewards are considered in RL, and the agent immediately switches to browsing.

Level of Supervisory Control: The discount factor γ_{type} approximates the level of executive control of individuals. Figure 5d illustrates the effect of high executive control: writing is performed uninterruptedly while inhibiting switches to tasks with higher immediate but lower long-term gains.

Comparison with Human Data

Novel experimental data was collected to assess (i) how well the model generalizes to an unseen task environment and (ii) if it can account for individual differences. The study was conducted on an online crowd-sourcing platform. Participants' data was only used if they completed a minimum of 6 trials, switched tasks within trials, and did not exceed or subceeded reasonable thresholds in trial times and attained rewards. Participants practiced each task type separately prior to entering interleaving trials. Six task instances were made available on a browser view. The

reward structure of each task was explained, and users had to decide how to maximize points within a limited total time. Again, the agent was trained for 250 episodes using the discounted reward HO-MAXQ algorithm (Andre and Russell 2002).

Method

Experimental Environment

The trials of the experiment were conducted on a web page presenting a task interleaving problem (see Fig. 6). Each interleaving trial consisted of six task instances of four different task types. The four different task types were math, visual matching, reading, and typing. Each task type consisted of different subtasks. All task instances were shown as buttons in a menu on the left side of the UI. Task instances were color coded according to their respective task type. The attended task was shown in a panel to the right of the task instances menu. Participants were informed about the score they attained in the current trial with a label on the top right. The label also showed the attained reward of the last subtask in brackets. For all task types, participants were allowed to leave a task at any point and were able to continue it at the position they have left it earlier. However, it was not possible to re-visit previously completed subtasks. Tasks for which all subtasks were completed could not be selected anymore. No new tasks were introduced into the experimental environment after a trial started.

Fig. 6 Reading task with text (top) and multiple-choice questions (bottom)

Round (1 of 5)

Calculating A

Calculating B

Checking A

Checking B

Reading A

Typing A

Reading A

Paragraph number 1

You cannot scroll up!

HIGH AVALANCHE DANGER IN SOME REGIONS
CURRENT CONDITIONS

From Monday evening until Tuesday morning, 10 to 20 cm of snow fell over a wide area on the northern flank of the Alps and in northern Grisons. Less snow fell elsewhere. There were some bright spells in Grisons. On the southern flank of the Alps, the weather was sunny. Midday temperatures ranged from minus 7 degrees in the north to minus 4 degrees on the southern flank of the Alps. The west to southwesterly wind was light to moderate, but strong at times. In the afternoon, further precipitation arrived from the northwest.

Since the precipitation began on Sunday evening, 20 to 30 cm of

Next paragraph

Where are natural avalanches possible?

☐ Other regions with a lot of fresh snow

☒ Inneralpine regions

☐ Northern Alpine regions

☒ Ticino

☐ Southern Alpine regions

How to evaluate snow drift accumulations?

☒ With caution

☐ With prudence

☐ With attention

☐ With care

☐ With wariness

Score: 9 (+5)

Tasks and Task Models

In this section, we explain the tasks of our experiment and how we designed the respective task models. In-task rewards were designed to be realistic and clear. Participants were told about the reward structures of tasks and how reward correlates to monetary reward (shown in table). The explanation of reward structures was held simple for all task types (e.g., “you receive 1 point for each correctly solved equation/answered question.”). Feedback on attained rewards was provided (see score label in Fig. 6). A mapping was created between what is shown on the display and task state s . Figure 7a illustrates this for the reading task where text paragraphs are mapped to the state of the reading model. Task models were used in the RL environment of the HRL agent. All tasks could be left in any state.

Reading: Reading tasks featured a text box on top, displaying the text of an avalanche bulletin, and two multiple-choice questions to measure text comprehension displayed in a panel below (see Fig. 6). The progress of participants within a reading task was tracked with the text’s scroll bar. After finish reading a paragraph, participants had to click the “Next paragraph” button to advance. The button was only enabled when the scroll bar of the current paragraph reached its end. Per correctly answered question participants attained ten points of reward.

Reading Model: An example of a reading task model is presented in Fig. 7b. Each state represents several lines of text of the avalanche bulletin. The bumps in the cost function $c_{T_r}(s)$ match with the end of paragraphs. The two states of the reward function $r_{T_r}(s)$ that provide ten points of

Fig. 7 Task models of the four tasks used in the experiment: **a** Example of how task state is assigned to visible state on display: passages of text in the reading task are assigned to the discrete states of its task model (column of numbers) over which reward (green) and cost

function (red) are specified. The row highlighted yellow provides the answer to a comprehension query at the end. Exemplary task models for **b** reading, **c** visual matching, **d** math, and **e** typing tasks

reward match the respective lines of the avalanche bulletin which provide an answer to one of the multiple-choice questions.

Visual Matching: Visual matching tasks featured a scrollable list of images (see Fig. 8). From these images participants had to identify those that display airplanes. This was done by clicking on the respective image. Per correct click, participants attained one point of reward. A visual matching task contained six of these lists and participants could proceed to the next one by clicking the “Next subtask” button. Again, this button was only enabled when the scroll bar reached its end. Progress was tracked using the scroll bar.

Visual Matching Model: An example of a visual matching task model is presented in Fig. 7c. Each state represents several images. The bumps in the cost function $c_{T_v}(s)$ depict the end of an image list. The number of points that is returned by the reward function $r_{T_v}(s)$ for a specific state s depends on the number of images in that state that are airplanes (1 point per plane).

Math: In math tasks, equations were displayed in a scrollable list (see Fig. 9). Thereby, one number or operator was shown at a time and the next one only was revealed when scrolling down. Participants received one point of reward for each correctly solved equation. A math task contained six equations. Participants could proceed by

clicking the “Next equation” button. The button was only enabled when the scroll bar of the current equation reached its end. Progress was logged via the position of the scroll bar.

Math Model: An example of a math task model is presented in Fig. 7d. Each state represents several numbers or operators. The states at which the cost function $c_{T_m}(s)$ returns zero represent the end of one of the six equations of a math task. Between ends the returned penalty of c_{T_m} increases linearly with the number of operators and numbers in equations. The reward function $r_{T_r}(s)$ returns one point of reward in the last state of each equation.

Typing: Typing tasks featured a sentence to copy at the top and a text box to type in below (see Fig. 10). Using HTML-functionality, we prevented participants to copy-paste the sentence into the text box. Participants received one point of reward for each correctly copied sentence. In a typing task, participants had to copy six sentences. Progress was tracked via the edit distance (Levenshtein 1966) between the text written by participants and the sentence to copy. They could proceed by clicking the “Next sentence” button that was enabled when the edit distance of the current sentence was zero.

Typing Model: An example of a typing task model is presented in Fig. 7e. Each state represents a discrete fraction of the maximal edit distance (Levenshtein 1966) of a

Fig. 8 Visual matching task with its scrollable list of images

Fig. 9 Math task with scrollable list showing operators and numbers (top) and text box to type in the result (bottom)

sentence to copy (capped at 1.0). The bumps in the cost function $c_{T_i}(s)$ match with the end of sentences. The reward function $r_{T_i}(s)$ provides a reward in the last state of each sentence.

Procedure

After instructions, informed consent, and task type specific practice, the participants were asked to solve a minimum of two-task interleaving trials but were allowed to solve up to five trials to attain more reward. Every trial contained six task instances, each sampled from a distribution of its general type. Trial durations were sampled from a random distribution unknown to the participant. The distribution was constrained to lie between 4 and 5 min. This limit was chosen empirically to ensure that participants cannot complete all task instances of a trial and are forced to interleave them to maximize reward. The stated goal was to maximize total points linked to monetary rewards. No

Fig. 10 Typing task with text to copy (top) and text box to type in (bottom)

task instance was presented more than once to a participant. The average task completion time was 39 min. The average number of completed task interleaving trials was 3.

Participants

218 participants completed the study. Ten were recruited from our institutions, and the rest from Amazon Mechanical Turk. Monetary fees were designed to meet and surpass the US minimum wage requirements. A fee of 5 USD was awarded to all participants who completed the trial, and an extra of 3 USD as a linear function of points attained in the interleaving trials. We excluded 7 participants who did not exhibit any task interleaving behavior or exceeded respectively subceeded thresholds in attained rewards or trial times.

Model Fitting

Empirical Parameters: Given the same set of tasks, humans choose different interleaving strategies to accomplish them. This can be attributed to personal characteristics like varying levels of executive control or a different perception of the resumption costs of a particular task (Janssen and Brumby 2015). In our method, we model individual differences with a set of personal parameters. More specifically, we introduce parameters that can scale the cost function of each task type and a parameter to model a constant cost that is paid for every switch. In this way, each cost function can be adjusted to model the perceived costs of an individual person. The personal cost function c_{PT} of a task T is defined as $c_{PT}(s) = c_P + s_{PT} c_T(s)$ where $0.0 < c_P < 0.3$ is a constant switch cost paid for each switch and $0.0 < s_{PT} < 1.0$ is a scaler of the task type's general cost function $c_T(s)$. In addition, we also fit γ_{type} , the discount factor of the task type hierarchy of our model to data ($0.0 < \gamma_{type} < 1.0$). γ_{type} is used to model various degrees of executive control.

Inverse Modeling Method: To fit these to an individual's data, we used *approximate Bayesian computation* (ABC) (Kangasrääsiö et al. 2017; Lintusaari et al. 2018). ABC is a sample-efficient and robust likelihood-free method for fitting simulator models to data. It yields a posterior distribution for the likelihood of parameter values given data. An aggregate index of interleaving similarity is the to-be-minimized discrepancy function:

$$d(S_S, A_S, S_I, A_I, \pi_{root}, \pi_{type}) = -w \left[2 - \left(\frac{1}{N_S} \sum_{s_S, a_S \in S_S, A_S} \mathbb{I}(a_S, \pi_{root}(s_S)) \right) + \frac{1}{N_I} \sum_{s_I, a_I \in S_I, A_I} \mathbb{I}(a_I, \pi_{type}(s_I)) \right] \quad (7)$$

$$\mathbb{1}(a_p, a_a) = \begin{cases} 1 & \text{if } a_p = a_a \\ 0 & \text{else,} \end{cases}$$

where S_s is the set of states in which participants switched tasks, A_s is the set of chosen actions (tasks), S_l are the states in which participants left a task, and A_l is the set of leave actions. N_s , N_l are the number of task switches respectively leave actions. π_{root} is the root-level policy of the HRL agent, and π_{type} is its type-level policy. Note that this accuracy metric collapses the *next task* and *leaving a task* accuracies reported in the paper.

Fitting Procedure: We held out the *last* trial of a participant for testing and used the preceding interleaving trials for parameter fitting. We run the above fitting method to this data for 60 iterations. In each, we trained the HRL agent ten times using the same set of parameters in a task interleaving environment matching that of the participant in question. For the Gaussian Process proxy model in ABC, we used a Matern-kernel parameterized for twice-differentiable functions. On a commodity desktop machine (Intel Core i7 4 GHz CPU), learning a policy took on average 10.3 sec (SD 4.0), and fitting for full participant data took 103.8 min (SD 28.2). The reported results come from the policy with lowest discrepancy to data obtained in 15 repetitions of this procedure with different weights (best: $w = 100$).

Baseline Models

To analyze the capability of our HRL model in terms of reproducing human task interleaving, we compared it against several variants of two other models: a flat RL agent and an omniscient-myopic agent. In total, our experiment had the following ten baseline models:

1. **HRL** chose actions according to the policy of our HRL model.
2. **HRL-Up** was our HRL agent trained on the test trial. As such, it is the upper bound of our HRL model as learned expected rewards match with the actual rewards that were attained in the test trial.
3. **HRL-Myopic** was the myopic version of our HRL model. A myopic model only considers the reward attainable in the next state for choosing an action. This was modeled by setting the discount factor to zero ($\gamma_{type} = 0$).
4. **RL** was a flat RL agent modeling human task interleaving. Its Bellman equation is defined as

$$\begin{aligned} Q_{flat}^{\pi}(s, a) = & \sum_{s', t, SA(s', t)} F_{flat}(t|s, a) \\ & P_{flat}(s'|s, a)[R_{flat}(s, s') \\ & + \gamma_{flat}^t Q_{flat}^{\pi}(s', \pi_{flat}(s'))], \end{aligned} \quad (8)$$

where s , a , s' , t , P_{flat} , F_{flat} , π_{flat} , and γ_{flat} are the respective functions or parameters of the flat RL agent. $SA(s', t)$ is a function that returns the subset of available next states s' and transition times t of other tasks in the environment. $R_{flat}(s, a, s')$ is the reward function of the flat RL agent and is defined as

$$R_{flat}(s, s') = \begin{cases} -c_T(s) - c_T(s') & \text{if } task(s) \neq task(s') \\ r_T(s) & \text{else,} \end{cases} \quad (9)$$

where $task$ is a function that returns the task of the respective state, $c_T(s)$ represents interruption and future resumption costs of state s , and $c_T(s')$ the resumption costs of the next state s' (see “[Reward Functions](#)” for more details). Using Q-learning, we trained the flat RL agent for 250 episodes, which was sufficient for the expected reward to saturate. For parameter fitting of the RL model, we define the following discrepancy function:

$$\begin{aligned} d(S_s, A_s, S_l, A_l, \pi_{flat}) = & -w \left[2 - \left(\frac{1}{N_s} \sum_{s_s, a_s \in S_s, A_s} \right. \right. \\ & \left. \left. \mathbb{1}(a_s, \pi_{flat}(s_s)) + \frac{1}{N_l} \sum_{s_l, a_l \in S_l, A_l} \right. \right. \\ & \left. \left. \mathbb{1}(a_l, \pi_{flat}(s_l)) \right) \right]. \end{aligned} \quad (10)$$

5. **RL-Up** is the upper bound of our RL model. It is trained like HRL-Up.
6. **RL-Myopic** is the myopic version of the RL model (with $\gamma_{flat} = 0$).
7. **Om.-Myopic** is an omniscient-myopic policy that chooses the task T that provides the highest reward in its next state s' :

$$\begin{aligned} \pi^{OM} = & \arg \max_T [r_T(s'_T) - \mathbb{1}(s'_T, s) [c_T(s'_T) + c_T(s)]], \\ \mathbb{1}(s', s) = & \begin{cases} 0 & \text{if } task(s') = task(s) \\ 1 & \text{else,} \end{cases} \end{aligned} \quad (11)$$

where s'_T is the next state of task T , s is the current state of the ongoing task, and c_T is the respective task's cost function. To compare against a strong model, it decides based on the *true* rewards and costs of the next states. By contrast, HRL decides based on learned *estimates*.

8. **Om.-Reward** is a variant of Om.-Myopic that only considers reward: $\pi^{OR} = \arg \max_T r_T(s'_T)$.
9. **Om.-Costs** is another variant of Om.-Myopic that only considers costs: $\pi^{OC} = \arg \max_T \mathbb{1}(s'_T, s) [c_T(s'_T) + c_T(s)]$.
10. **Random** chooses at each decision point of the SMDP one of the available actions at random.

Myopic models only consider the reward attainable in the next state in their task switching decisions and, hence, tend to switch to tasks with immediate gratification. Intuitively, these models would switch to the browsing task in the example of Fig. 1 as soon as there is no higher immediate reward available in writing (states 0–2 and 5–7). All omniscient models possess a myopic reward horizon. However, rather than deciding on estimated expected rewards, they know the actual reward (and/or cost) in the next state and decide based on it. RL- and HRL-Up can be considered omniscient models as they are trained on the task environment of the test trial. In contrast to the other myopic models, they consider time-discounted future rewards when deciding for which task to attend. Considering the example of Fig. 1, they could exert behavior where task switches to the browsing task are inhibited to attain the large delayed reward of writing (states 8–9). In general, RL and HRL models differ in that HRL conducts two decisions to switch between tasks (decision 1: leave the current task; decision 2: task to attend to next) while in RL a task switch is a single decision (see “Hierarchical Decomposition of Task Environments”).

All HRL and flat RL models were fitted to the data of individual participants using the model fitting procedure described in “Model Fitting.” We did not compare against marginal rate of return (Duggan et al. 2013) or information foraging models (Payne et al. 2007) as in-task states can have zero reward. Both models would switch task in this case, rendering them weaker baselines than Om.-Myopic. The multi-criteria model of Wickens et al. (2015) does not adapt to received task rewards and offers no implementation. Models of concurrent multitasking (i.e., Oberauer and Lewandowsky 2011; Salvucci and Taatgen 2008) are not designed for sequential task interleaving.

Results

Predictions of HRL were made for the held-out trial and compared with human data. Analyzing base rates for continuing versus leaving a task of the behavioral sample revealed that task continuation dominates events ($= 0.95$). For this reason, we analyze the capability of models to predict if participants leave or continue a task separately. As normality assumptions are violated, we use Kruskal-Wallis for significance testing throughout. Pairwise comparisons are conducted using Tukey’s post hoc test

Empirical Data: Before comparing the performance of the various models, we inspected participant data. The most popular task type was visual matching which was selected by 95% of participants in one of their interleaving trials (see Fig. 11). It was followed by math (78.5%), typing (70.0%),

Fig. 11 Fraction of participants that selected task instances of a particular type in an interleaving trial

and reading tasks (65.5%). The unconditioned probability distribution $E_T(t)$ of logged state transition times per task type over all participants shows that these differ between the task types of our study (see Fig. 12). Participants seem to be faster in transitioning between states in reading and visual matching tasks compared with math and typing tasks. We use $E_T(t)$ to approximate $F(t|s, a)$ in Eq. 1 when training our HRL agent (see “Modeling State Transition Times”).

Reward: Participants attained a mean reward of 33.18 (SD 11.92) in our study (see Fig. 13 a). Om.-Costs attained the lowest difference in reward compared with participants (M 34.44, SD 7.86), followed by Om.-Reward (M 33.85, SD 7.70), HRL-Myopic (M 34.45, SD 13.79), RL-Myopic (M 34.48, SD 14.26), Om.-Myopic (M 34.72, SD 8.11), and HRL (M 36.61, SD 9.35). Higher differences in reward compared with participants are attained by HRL-Up (M 38.41, SD 9.71), RL (M 39.11, SD 9.51), RL-Up (M 39.66, SD 12.35), and Random (M 20.5, SD 8.96). Differences are significant ($H(10) = 332.5, p < 0.001$) and a pairwise comparison indicates that this holds for all comparisons with random ($p < 0.001$) as well as for the comparisons of Participants with HRL-Up ($p < 0.001$), RL ($p < 0.001$), and RL-Up ($p < 0.001$).

Choosing Next Task: HRL-Up showed the highest accuracy in predicting the next task of a participant (M 0.55, SD 0.27) (see Fig. 13 b). It was followed by HRL (M 0.5, SD 0.27), Om.-Costs (M 0.43, SD 0.26), Om.-Myopic (M 0.41, SD 0.26), RL-Up (M 0.4, SD 0.23), Om.-Reward (M 0.4, SD 0.25) and RL (M 0.38, SD 0.23). A lower accuracy was attained by HRL-Myopic (M 0.26, SD 0.28), RL-Myopic (M 0.28, SD 0.23), and Random (M 0.24, SD 0.21). There was a significant effect of model ($H(9) = 270.5, p < 0.001$). Tukey’s test indicated a significant difference between all baseline models and HRL-up ($p < 0.001$) as well as Random ($p < 0.001$). HRL performed significantly better than RL models (RL-Up: $p = 0.01$, RL $p < 0.001$) as well as Om.-Myopic ($p = 0.04$) and Om.-Reward ($p = 0.02$). Myopic versions of HRL and RL

Fig. 12 Unconditioned probability distribution $E_T(t)$ of logged state transition times per task type over all participants. y-axis is probability and x-axis is transition time

performed significantly worse than all other baseline except Random ($p < 0.001$ for all).

Leaving a Task: HRL-Up outperformed all other baseline models in predicting when a participant would leave a task (M 0.94, SD 0.13). It is followed by HRL (M 0.85, SD 0.23), RL-Up (M 0.83, SD 0.24), HRL-Myopic (M 0.82, SD 0.27), Om.-Costs (M 0.78, SD 0.27), Om.-Reward (M 0.76, SD 0.28), RL (M 0.75, SD 0.3), Om.-Myopic (M 0.75, SD 0.29), and RL-Myopic (M 0.68, SD 0.34). Random was the worst (M 0.51, SD 0.29) (see Fig. 13 c). Differences between models are significant ($H(9) = 279$, $p < 0.001$). A pairwise comparison indicates that this holds for all comparisons with Random ($p < 0.001$). HRL-Up performs significantly better than all baseline models ($p \leq 0.001$) save HRL. HRL performs significantly better than all other baselines ($p < 0.01$) except Om.-Costs and RL-Up.

Continuing a Task: RL-Myopic (M 0.97, SD 0.06) was better than RL (M 0.93, SD 0.11) and RL-Up (M 0.92, SD 0.11) in predicting continuation in a task (see Fig. 13 d). HRL models followed, with HRL-Up (M 0.92, SD 0.1)

outperforming HRL (M 0.86, SD 0.17) and HRL-Myopic (M 0.85, SD 0.1). Omniscient-myopic models attained a lower accuracy, with Om.-Myopic (M 0.73, SD 0.22) performing better than Om.-Reward (M 0.72, SD 0.23) and Om.-Costs (M 0.71, SD 0.22). Random was the worst (M 0.51, SD 0.06). These differences were significant ($H(9) = 956.3$, $p < 0.001$). Pairwise comparisons indicated that there are no significant differences between omniscient-myopic models as well as for HRL-Up and HRL. The same is true for the comparisons of HRL-Myopic with RL and RL-up as well as for RL with RL-Up and RL-Myopic. All other pairwise differences are significant ($p < 0.001$).

Order of Tasks: We define *task order error* as the sum of non-equal instances between produced orders of tasks of a model with the respective participant (see Appendix 2 for details). A significant omnibus effect of model was found ($H(9) = 533.2$, $p < 0.001$). Om.-Costs had the smallest error (M 20.01, SD 27.1) followed by Om.-Myopic (M 20.54, SD 28.96), Om.-Reward (M 20.82, SD 28.11), RL (M 22.79, SD 30.18), RL-Up (M 23.13, SD 32.07), RL-Myopic (M 25.39, SD 34.88), HRL-Myopic (M 25.54, SD

Fig. 13 Means and 95% confidence intervals for **a** attained rewards (significance notation with respect to participants), **b** accuracy in predicting next task, **c** accuracy in predicting leaving of a task, **d** accuracy

in predicting continuing of a task, and **e** error in predicting order of tasks (lower is better). For **b–e**, significance notation is with respect to HRL

33.1), HRL (M 26.22, SD 30.14), and HRL-Up (M 28.85, SD 35.32). However, these differences were not statistically significant. Random was the worst (M 182.47, SD 124). All models had a significantly smaller error than Random ($p < 0.001$ for all).

State Visitations: We computed histograms of state visitation frequencies per task type (see Fig. 14). As visual inspection confirms, HRL-Up (0.95) and HRL (0.93) had a superior histogram intersection with participants than other baseline models. They were followed by RL-Up (0.92), RL (0.91), Om.-Costs (0.89), Om.-Reward (0.89), HRL-Myopic (0.88), RL-Myopic (0.88), Om.-Myopic (0.80) and Random (0.81). The step-like patterns in the histograms of Participants were reproduced by HRL and RL models, illustrating that its policies switched at the same subtask boundaries as participants (e.g., see top-row in Fig. 14). However, the histograms of HRL models show a higher overlap with participants' histograms than RL models.

Comparison with Flat RL

To better understand the implications of hierarchicality in the context of task interleaving, we further compared our HRL model with the flat RL implementation. Thus, we learned 100 policies for a ten task, six instance problem and the same simulated user using default values for cost scalars (c_P and s_{PT}) and γ_{type} . Figure 15 shows the learning curves of the two methods. HRL converged faster than flat RL which is in line with prior work (Dietterich 1998; Andre and Russell 2002; Ghavamzadeh and Mahadevan 2002). This is

Fig. 15 Learning curves of the flat RL and our HRL agent. Solid line denotes mean reward (y-axis) per episode (x-axis). Shaded area represents standard deviation

due to a significant decrease in the number of states (43-fold for this example). It is important to note that the optimal policy of flat RL and HRL for a given problem are the same. This experiment exemplified this, as both perform similarly in terms of attained reward after convergence.

Parameter Fitting

Table 1 reports the mean fraction of reproduced actions per participant for each iteration of our model fitting procedure. Fractions are computed using the normalized sum of reproduced actions of Eq. 7. Results on training trials improve with each iteration of the procedure and show that learned parameters generalize to the held-out test trials.

Fig. 14 State visitations: HRL shows better match with state visitation patterns than Myopic and Random. y-axis shows fraction of states visited aggregated over all trials

Table 1 Mean and standard deviation of fractions of reproduced actions of participants. Fractions are computed for fitted (test trial), random and inferred parameters after number of iterations (training trial)

Trials	Iterations	Mean	Std. dev.
Training	Random	0.51	0.21
	1	0.64	0.13
	10	0.69	0.13
	30	0.70	0.14
	60	0.73	0.13
Test	Random	0.54	0.21
	Fit	0.67	0.20

The mean difference between the estimated parameters of two runs of the fitting procedure per participant are: γ_t (discount factor) M 0.20, SD 0.23; c_P (switch cost) M 0.09, SD 0.09; s_{PR} (reading) M 0.32, SD 0.28; s_{PV} (visual matching) M 0.36, SD 0.28; s_{PM} (math) M 0.33, SD 0.26; s_{PT} (typing) M 0.38, SD 0.34. The somewhat low reliability of parameters can be explained by the fact that, in our task, participants can achieve a high reward by two means: high switch costs or high discount factor. While our model parameters are theoretically justified, refining the model such that parameter values can reliably be associated with behavioral patterns is an interesting direction of future work.

Model Inspection

To further inspect the performance of our HRL agent, we compared interleaving sequences of individual participants with those reproduced by the agent for the particular participant. Figure 16 shows the interleaving sequences produced by the HRL agent which attained (b) the lowest, (c) the closest to average, and (d) the highest error in task order compared with the sequences of the respective participant (according to Eq. A2.1). The interleaving sequence with the lowest error reproduces the order of task types of participants almost exactly. In contrast, the interleaving sequence of the closest to average and highest error task order between participant and agent are interchanged. However, both of these participants exhibit a lot of task switches and conduct particular tasks without attaining points for it.

Figure 17 shows the state-action value functions ($Q(s, a)$, see Eq. 1) for the different levels and task types of our HRL agent trained by using the optimal parameters of one participant of our study. On the task type level, the pattern of the state value of the action *Continue* matches with the reward function of the respective task type (see Fig. 7). The same holds for the action *Leave* and the cost function of a task type. The state-action value functions on root level of our HRL agent ($Task_{NN}$) show the expected reward for entering a task type instance at a particular state. These

values have converged to approximate the sum of expected rewards of *Continue* and *Leave* actions of the task type level.

Discussion

The results of our study provided evidence for hierarchical reinforcement learning (HRL) as a mechanism in task interleaving. Considering all metrics, it reproduced human task interleaving significantly better than all other baselines. In particular, state visitation histograms show that HRL exhibits more human-like behavior in terms of leaving at certain subtask boundaries and avoiding to continue tasks with low gratification.

The omniscient-myopic models proved to be strong baselines as they managed to reproduce human task interleaving behavior on most metrics. Interestingly, the model that only considers costs (Om.-Costs) better reproduced participant behavior than the omniscient models that considered rewards and costs (Om.-Myopic) or only rewards (Om.-Rewards). This indicates that humans may prioritize avoiding cognitive costs when making task switching decisions over gaining rewards. Given our study setting, this intuitively makes sense as costs needed to be certainly paid while reward was not necessarily gained (e.g., when answering questions wrong). This finding is also inline with related work that revealed the human tendency to push task switches to task and subtask bounds (Altmann and Trafton 2002; Janssen et al. 2012; McFarlane 2002). However, all omniscient-myopic models have the tendency to leave a task when participants still continued with it. In contrast, HRL and RL models reproduce this factor of participant behavior significantly better. This highlights the necessity of task interleaving models to consider long-term rewards in tasks to model human executive control.

The importance of considering long-term rewards not only within the task but also to choose the correct next task is indicated in comparing the accuracy of reproduced task switches of HRL models that consider

Fig. 16 Examples of reproduced task interleaving sequences of the HRL agent with respect to a particular participant. **a** Task models of the reproduced trial. Task interleaving sequences of participant and HRL agent with the **b** lowest, **c** closest to average, **d** highest task order error according to Eq. A2.1

discounted future rewards within tasks (HRL, HRL-Up) and HRL-Myopic. The former models choose next tasks that match significantly better with participants' selections than myopic HRL. To understand if our model can generalize task type knowledge to previously not encountered instances, we compare the results of HRL and HRL-up. HRL-Up was learned and tested on the test trial of our study and, hence, has encountered the instances it is tested on during training. No significant differences between HRL and HRL-Up on all metrics show that our model can indeed generalize to unobserved task instances.

All models performed worse when predicting participants' next task than when predicting whether they continue or leave the current one. This can be partly explained by

the fact that the first problem is harder, Random was at 0.24, than the latter two binary predictions where Random was at 0.51. In addition, we assume that this discrepancy comes from our reward function not capturing all factors that affect the decision to attend a particular task. Research found that the attributes of task difficulty, task priority, task interest, and task salience influence humans' task decisions (Wickens et al. 2015). Our reward function only models priority defined by the attainable points per task, deciding on participants' monetary reward. While task salience and interest can be assumed to be the same among all our task types, their task difficulty differs significantly. This difference affected participants' choices in our experiment. For example, task instances of type reading gave the most

Fig. 17 State-action value function $Q(s, a)$ of the HRL agent of one participant. *Continue* and *Leave* shows the expected reward per state for the respective action on task type level. $Task_{NN}$ shows the expected reward for entering a particular task type on root level

reward and still were the least popular, possibly due to their difficulty (see Fig. 11). Not modeling task difficulty in the reward function caused the low prediction accuracy in participants' next task decisions. In contrast, it did not affect the prediction of whether to leave or to continue a task as these decisions are primarily made to avoid switching costs. In future work it will be interesting to consider the identified task attributes of Wickens et al. (2015) in the reward functions of task types. As factors such as task difficulty and interest differ from human to human, we will use ABC to fit such a reward function to participant data. Such a setting would allow to adapt to individual differences in the perception of task reward. Furthermore, to model real-world task interleaving problems, it will be interesting to learn entirely unknown reward and cost functions of tasks from human demonstrations (similar to (Krishnan et al. 2016)).

Comparing the RL models with all other baselines shows that they perform best in terms of attained reward. However, they do not match with participant behavior as good as the HRL- and the omniscient-myopic models. Results indicate that RL exhibits super human performance even after fitting its parameters to participant data. As converged policies of HRL and flat RL should perform the same (see “Comparison with Flat RL”), this difference can only be explained by the interplay of the ABC fitting procedure with the RL- respectively HRL model. HRL disentangles the current task from the next task in its model (see “Decision Processes”). Thus, ABC can consider the lower and higher level policy separately in the discrepancy function (see Eq. 7) and, hence, identify differences in perceived costs per task type and participant. In contrast, RL learns expected reward for a task switch over the current and the next task. Thus, it cannot distinguish between costs

that are induced by the current task from costs that are induced by the next task in the discrepancy function (see Eq. 10). As a result, ABC cannot successfully identify perceived costs per participant and task type for the flat RL model. Thus, ABC's ability to identify the correct perceived costs of task types depends on the difference of how HRL and flat RL model task switching. Therefore, we conclude that the significantly better match of HRL models with participant behavior than RL models might indicate that humans hierarchically abstract task environments when learning decisions strategies for task interleaving. However, future research is necessary to confirm this hypothesis.

Participants of our study continued tasks at a much higher rate (95%) than reported in literature (60%, (Wickens et al. 2015)). This can be explained by the fact that in previous studies participants were asked whether they want to continue or switch to another task. In contrast, we assume participants to decide to stay within a task each time a state transition according to a task model is logged. The large number of states in these models explains the high task continuation rate of our study. In addition, the tasks of our study were more difficult than the analyzed study tasks of Wickens et al. (e.g., judgment tasks) which has shown to increase task continuation (Gutzwiller 2014).

In this paper, we have assumed that instead of learning policies per task instance, people learn expected rewards and costs per task type. This assumption is cognitively plausible, because it drastically reduces the amount of parameters, and because it allows generalizing past experience to previously unseen instances. How people recognize tasks as instances of types and how these expectations are generalized, is an exciting question for future work.

Conclusion

In this paper, we propose a theoretically justified hierarchical reinforcement learning model of human task interleaving. It overcomes the shortcomings of the previous state of the art by being able to adapt to multiple tasks and complex reward/cost structures. In addition, it can consider non-diminishing, delayed rewards for task switching decisions, which are common in many day-to-day activities. The model assumes a two-level supervisory control system and both levels approximate utility based on experience. The higher level keeps track of ongoing tasks and decides which task to conduct next. The lower level decides based on the expected reward within a task whether to continue. The hierarchically optimal decomposition of the RL-task interleaving problem enables our lower level to consider the expected rewards of all other available tasks in its decision.

This model has demonstrated to be capable to reproduce common phenomena of task interleaving like sensitivity to costs of resumption (Altmann and Trafton 2002; Gutzwiller et al. 2019; Iqbal and Bailey 2008) and the human tendency to push switches to task boundaries where switch costs are lower (Altmann and Trafton 2002; McFarlane 2002). In addition, the HRL agent proved to be sensitive to in-task rewards (Horrey and Wickens 2006; Iani and Wickens 2007; Norman and Shallice 1986; Wickens and McCarley 2008) even when gratification was delayed. This allows the model to depict varying levels of executive control of individuals. These results corroborate emerging evidence for hierarchical reinforcement learning in supervisory control (Botvinick 2012; Frank and Badre 2011).

Our study has provided new evidence for hierarchical reinforcement learning (HRL) as a model of task interleaving. The resemblance between simulated and empirical data is very encouraging. Comparison against myopic baselines suggests that human interleaving is better described as optimal planning under uncertainty than by a myopic strategy. We have shown that hierarchically optimal value decomposition is a tractable solution to the planning problem that the supervisory control system faces. In particular, it (i) can achieve a high level of control via experience, (ii) adapts to complex and delayed rewards/costs, avoiding being dominated by immediate rewards, and (iii) can generalize task type knowledge to instances not encountered previously. Moreover, only a small number of empirical parameters was needed for characterizing individual differences.

One exciting remaining question is if humans indeed learn task interleaving from experience by hierarchically decomposing the computation of when to attend a new task and which task to attend. While our results provide first indications that this might be the case, further research is necessary to pinpoint the computational mechanism. In particular, it would be interesting to investigate if humans

need to observe a particular task type to task type switch to compute a meaningful expectation of gratification or if, due to hierarchical abstraction, humans do not consider the current task when computing expectations of in-task rewards of next tasks.

Another promising direction concerns modeling further cognitive factors that are involved in human task interleaving. We model recall effort to be the sole source of switch costs. In future work, it is interesting to extend cost functions to account for other factors, e.g., cognitive load. Furthermore, we will model task interleaving as a non-stationary (restless) problem, where memory traces decay during processing of other tasks and require more attentional refreshing the longer they were not accessed (Monk et al. 2004; Oberauer and Lewandowsky 2011). Finally, it is interesting to model increased switching costs between similar tasks caused by interferences of recall structures in working memory (Edwards and Gronlund 1998; Kiesel et al. 2010).

Funding Open access funding provided by Swiss Federal Institute of Technology Zurich. This work was funded in parts by the Swiss National Science Foundation (UFO 200021L_153644).

Data Availability Collected data of crowd-sourcing task interleaving study will be published after acceptance.

Code Availability Code of model will be published after acceptance

Compliance with Ethical Standards

Conflict of Interest

- Andrew Howes, University of Birmingham
- Duncan P. Brumby, University College London
- Christian P. Janssen, Utrecht University
- James Hillis, Facebook Reality Labs
- Bas van Opheusden, Princeton University

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix 1. Learning a hierarchically optimal policy

For model-free training of our HRL agent, we use the discrete-time discounted reward HO-MAXQ algorithm

proposed in Andre and Russell (2002). Algorithm 1 shows its pseudo code.

Algorithm 1 A model-free discrete-time hierarchically optimal discounted reward RL algorithm (Andre and Russell 2002).

```

function DR-HO-MAXQ(Task  $i$ , State  $s$ )
  let  $seq = \{\}$  be the sequence of states visited while
  executing  $i$ 
  if  $i$  is a primitive action then
    execute action  $i$  in state  $s$ , observe state  $s'$  and reward
     $R(s', s, i)$ 
     $Q_{t+1}^r(i, s) \leftarrow R(s', s, i)$ 
    push state  $s$  into the beginning of  $seq$ 
  else
    while  $i$  has not terminated do
      choose action  $a$  according to behavioral policy
       $\pi_i(s)$ 
      let  $childSeq = \text{DR-HO-MAXQ}(a, s)$ , where
       $childSeq$  is the sequence of states visited while
      executing action  $a$ 
      observe result state  $s'$ 
      let  $a^* = \underset{a' \in A_i(s')}{\text{argmax}} [Q_t^r(s', a') +$ 
       $Q_t^c(i, s', a')]$ 
      for each  $s$  in  $childSeq$  do
         $Q_{t+1}^c(i, s, a) \leftarrow [Q_t^c(i, s', a^*) + Q_t^r(a^*, s')]$ 
         $Q_{t+1}^e(a, s, a'') \leftarrow$ 
         $\underset{a' \in A_i(s')}{\text{argmax}} Q_t(i, s', a')$ 
      end for
      append  $childSeq$  onto the front of  $seq$ 
       $s = s'$ 
    end while
  end if return  $seq$ 
end function

```

$Q_t^c(i, s, a)$ and $Q_t^e(i, s, a)$ specify the respective part of the value function decomposition at time t and hierarchy level i . $Q^t(i, s, a)$ is the overall state action value function at time t and on level i . a'' is the subtask of action a that is taken in state s .

Appendix 2. Task order error

To calculate the task order error, we execute model policies in the task environments of participants' test trials. The task order error is the distance between a model's and a participant's task interleaving sequence:

$$D_{i,j} = \begin{cases} ||i - j|| & \text{if } u_i = v_j \\ |u| & \text{if } v_j \text{ not in } u \\ 0 & \text{else,} \end{cases} \quad \text{for } 0 \leq i \leq |u|, 0 \leq j \leq |v| \quad (\text{A2.1})$$

where u is the sequence of tasks of the participant and v the sequence of tasks of the respective model. Note that by considering Eq. A2.1 in the discrepancy function of the parameter fitting procedure, the performance of the HRL agent with respect to this metric can be improved. However, this is a trade-off as with this additional discrepancy measure the agent performs worse in the other reported metrics.

References

- Altmann, E., & Trafton, J. (2002). Memory for goals: an activation-based model. *Cognitive science*, 26(1), 39–83.
- Altmann, E., & Trafton, J. (2007). Timecourse of recovery from task interruption: data and a model. *Psychon Bull Review*, 14(6), 1079–1084.
- Andre, D., & Russell, S. (2002). State abstraction for programmable reinforcement learning agents. In *Eighteenth National Conference on Artificial Intelligence*, 119–125.
- Bailey, B., & Konstan, J. (2006). On the need for attention-aware systems: measuring effects of interruption on task performance, error rate, and affective state. In *Computers in Human Behavior*, (Vol. 22 pp. 685–708).
- Balaguer, J., Spiers, H., Hassabis, D., Summerfield, C. (2016). Neural mechanisms of hierarchical planning in a virtual subway network. *Neuron*, 90(4), 893–903.
- Botvinick, M. (2012). Hierarchical reinforcement learning and decision making. *Curr Opin Neurobiol*, 22(6), 956–962.
- Botvinick, M., Niv, Y., Barto, A. (2009). Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, 113(3), 262–280.
- Brumby, D., Janssen, C., Kujala, T., Salvucci, D. (2018). Computational models of user multitasking, pp. 341–362.
- Brumby, D., Salvucci, D., Howes, A. (2009). Focus on driving: how cognitive constraints shape the adaptation of strategy when dialing while driving. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1629–1638.
- Dietterich, T. (1998). The maxq method for hierarchical reinforcement learning. In *ICML*, 98, 118–126.
- Duggan, G., Johnson, H., Sørli, P. (2013). Interleaving tasks to improve performance: users maximise the marginal rate of return. *Int J Hum-Comput St*, 71(5), 533–550.
- Edwards, M., & Gronlund, S. (1998). Task Interruption and its Effects on Memory. *Memory*, 6(6), 665–687.
- Frank, M., & Badre, D. (2011). Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: computational analysis. *Cerebral Cortex*, 22(3), 509–526.
- Gershman, S.J., & Uchida, N. (2019). Believing in dopamine, nature reviews neuroscience, 1–12.
- Ghavamzadeh, M., & Mahadevan, S. (2002). Hierarchically optimal average reward reinforcement learning. In *ICML* (pp. 195–202).
- Gutzwiller, R. (2014). Switch choice in applied multi-task management, Ph.D. thesis, Colorado State University. Libraries.
- Gutzwiller, R., Wickens, C., Clegg, B. (2019). The role of reward and effort over time in task switching. *Theoretical Issues in Ergonomics Science*, 20(2), 196–214.
- Horrey, W., & Wickens, C. (2006). Examining the impact of cell phone conversations on driving using meta-analytic techniques. *Human factors*, 48(1), 196–205.
- Iani, C., & Wickens, C. (2007). Factors affecting task management in aviation. *Human factors*, 49(1), 16–24.

- Iqbal, S., & Bailey, B. (2008). Effects of intelligent notification management on users and their tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 93–102). <https://doi.org/10.1145/1357054.1357070>.
- Janssen, C., & Brumby, D. (2010). Strategic adaptation to performance objectives in a dual-task setting. *Cognitive science*, 34(8), 1548–1560.
- Janssen, C., & Brumby, D. (2015). Strategic adaptation to task characteristics, incentives, and individual differences in dual-tasking. *PLOS ONE*, 10(7), 1–32.
- Janssen, C., Brumby, D., Garnett, R. (2012). Natural break points: the influence of priorities & cognitive & motor cues on dual-task interleaving. *J. Cogn. Eng. Decis. Mak.*, 6(1), 5–29.
- Jersild, A. (1927). Mental set and shift. *Arch. of psychology*.
- Kaelbling, L., Littman, M., Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1–2), 99–134.
- Kangasrääsio, A., Athukorala, K., Howes, A., Corander, J., Kaski, S., Oulasvirta, A. (2017). Inferring cognitive models from data using approximate Bayesian computation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (pp. 1295–1306).
- Kangasrääsio, A., Jokinen, J.P., Oulasvirta, A., Howes, A., Kaski, S. (2019). Parameter inference for computational cognitive models with approximate Bayesian computation. *Cognitive science*, 43(6), e12738.
- Kiesel, A., Steinhauser, M., Wendt, M., Falkenstein, M., Jost, K., Philipp, A.M., Koch, I. (2010). Control and interference in task switching—a review. *Psychological Bulletin*, 136(5), 849–874.
- Kriegeskorte, N., & Douglas, P. (2018). Cognitive computational neuroscience. *Nature Neuroscience*, 21(9), 1148–1160.
- Krishnan, S., Garg, A., Liaw, R., Miller, L., Pokorny, F.T., Goldberg, K. (2016). Hirl: hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards. [arXiv:1604.06508](https://arxiv.org/abs/1604.06508).
- Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10, 707–710.
- Lintusaari, J., Vuollekoski, H., Kangasrääsio, A., Skytén, K., Järvenpää, M., Marttinen, P., Gutmann, M., Vehtari, A., Corander, J., Kaski, S. (2018). Elfi: Engine for likelihood-free inference. *JMLR*, 19(1), 643–649.
- McFarlane, D. (2002). The scope and importance of human interruption in human-computer interaction design. *Human-Computer Interaction*, 17(1), 1–61.
- Monk, C., Boehm-Davis, D., Mason, G. (2004). Recovering from interruptions: implications for driver distraction research. *Human factors*, 46(4), 650–663.
- Monsell, S. (2003). Task switching. *Trends in cognitive sciences*, 7(3), 134–140.
- Norman, D., & Shallice, T. (1986). Attention to action. *Consciousness and Self-Regulation: Advances in Research and Theory* Volume 4 (pp. 1–18).
- Oberauer, K., & Lewandowsky, S. (2011). Modeling working memory: a computational implementation of the Time-Based Resource-Sharing theory. *Psychon Bull Review*, 18(1), 10–45.
- Oulasvirta, A., & Saariluoma, P. (2006). Surviving task interruptions: investigating the impl. of long-term working memory theory. *Int J Hum-Comput St*, 64(10), 941–961.
- Payne, S., Duggan, G., Neth, H. (2007). Discretionary task interleaving: heuristics for time allocation in cognitive foraging. *Journal of Experimental Psychology: General*, 136(3), 370.
- Raby, M., & Wickens, C.D. (1994). Strategic workload management and decision biases in aviation. *The International Journal of Aviation Psychology*, 4(3), 211–240.
- Rasmussen, D., Voelker, A., Eliasmith, C. (2017). A neural model of hierarchical reinf. learning PloS one, 12 7.
- Rubinstein, J., Meyer, D., Evans, J. (2001). Executive control of cognitive processes in task switching. *Journal of Experimental Psychology: Human Perception and Performance*, 27(4), 763.
- Salvucci, D., & Taatgen, N. (2008). Threaded cognition: an integrated theory of concurrent multitasking. *Psychology Review*, 115(1), 101.
- Salvucci, D., Taatgen, N., Borst, J. (2009). Toward a unified theory of the multitasking continuum: From concurrent performance to task switching, interruption, and resumption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1819–1828).
- Sutton, R., & Barto, A. (1998). Introduction to reinforcement learning, vol. 135.
- Trafton, J., Altmann, E., Brock, D., Mintz, F. (2003). Preparing to resume an interrupted task: effects of prospective goal encoding and retrospective rehearsal. *Int J Hum-Comput St*, 58(5), 583–603.
- Wickens, C., Gutzwiller, R., Santamaria, A. (2015). Discrete task switching in overload: a meta-analyses and a model. *Int J Hum-Comput St*, 79, 79–84.
- Wickens, C., & McCarley, J. (2008). Executive control: attention switching, interruptions, and task management. In *Consciousness and self-regulation*, 145–160.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.