



Article

Mobile User Interface Adaptation Based on Usability Reward Model and Multi-Agent Reinforcement Learning

Dmitry Vidmanov *  and Alexander Alfimtsev

Information Systems and Telecommunications, Bauman Moscow State Technical University,
105005 Moscow, Russia; alfirm@bmstu.ru

* Correspondence: vidmanov@bmstu.ru

Abstract: Today, reinforcement learning is one of the most effective machine learning approaches in the tasks of automatically adapting computer systems to user needs. However, implementing this technology into a digital product requires addressing a key challenge: determining the reward model in the digital environment. This paper proposes a usability reward model in multi-agent reinforcement learning. Well-known mathematical formulas used for measuring usability metrics were analyzed in detail and incorporated into the usability reward model. In the usability reward model, any neural network-based multi-agent reinforcement learning algorithm can be used as the underlying learning algorithm. This paper presents a study using independent and actor-critic reinforcement learning algorithms to investigate their impact on the usability metrics of a mobile user interface. Computational experiments and usability tests were conducted in a specially designed multi-agent environment for mobile user interfaces, enabling the implementation of various usage scenarios and real-time adaptations.

Keywords: deep learning; reinforcement learning; multi-agent systems; adaptive systems; mobile user interface; usability; user experience



Citation: Vidmanov, D.; Alfimtsev, A. Mobile User Interface Adaptation Based on Usability Reward Model and Multi-Agent Reinforcement Learning. *Multimodal Technol. Interact.* **2024**, *8*, 26. <https://doi.org/10.3390/mti8040026>

Academic Editors: Jan Auernhammer, Takumi Ohashi, Di Zhu, Kuo-Hsiang Chen and Wei Liu

Received: 21 February 2024

Revised: 21 March 2024

Accepted: 22 March 2024

Published: 24 March 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The development of an innovative technology for automatic predictive adaptation of user interfaces for smartphones, tablets, and other mobile devices based on algorithms, models, and methods of deep machine learning is a relevant task. Currently, there is a demand in the IT industry to reduce the complexity of mobile app development, optimize computations in interface rendering, and improve usability in solving user tasks [1,2]. On average, a typical user installs about 100 applications on their mobile device but uses no more than 50 of them in a month and less than 10 apps every day [3,4]. It is assumed that predictive adaptation technologies for mobile user interfaces will significantly reduce the redundancy of applications, functions, and data and greatly increase the speed of information processing by the user [5,6].

In general, adaptivity can be defined as the capability of a system to achieve its goals in a changing environment, by selectively executing and switching between behavior models [7]. An adaptive user interface is an interface that can autonomously adjust the content, layout, or style to better suit the user's capabilities and interests [8]. Usually, automatic adaptive processes optimize the system state under conditions of limited resources [9]. The adaptation of a mobile user interface using a simple algorithm that adjusts the positions of elements based on the frequency of use shows a dependence of adaptation frequency on training epochs, like the existing law of evolutionary adaptation in nature [10]. Research in this direction using reinforcement learning technologies suggests that an intelligent agent based on deep machine learning models can acquire similar adaptive properties to those found in living organisms [11].

In reinforcement learning theory, intelligent agents learn to optimize their control over the environment [12]. An agent interacts with the environment through a sequence of

observations, actions, and rewards. Deep neural networks can be used to approximate the action value function, representing the maximum sum of discounted rewards at each time step achievable through some agent behavior policy [13].

Various reinforcement learning approaches are applied to enhance human–machine interaction. For instance, single-agent approaches, where the interface agent adapts the entire system content, including elements, widgets, and menus [14] or multi-agent approaches where the user-agent and interface-agent interact to improve cooperation when changing the user interface content [15], as well as multi-agent approaches with multiple widget agents—UI elements participating in adaptation [16]. It is natural to view interface elements as multiple agents adapting to different usage scenarios [17]. Therefore, this work utilizes the Multi-Agent Reinforcement Learning (MARL) approach to simultaneously train multiple agents.

The goal of the agent is to find a policy for optimal actions that maximizes cumulative rewards. Such rewards are called external and are generated by the environment surrounding the agent or defined as a model using expert knowledge about the task [18]. A typical study examines the effectiveness of creating reward models in the context of adapting a user interface using RL and a reward model obtained on the basis of predictive human–computer interaction, augmented with human feedback [19,20].

This paper proposes a usability reward (UR) model in multi-agent reinforcement learning to evaluate the action values of agents based on environmental and agent state parameters in a mobile user interface. The reward model can include various components, such as the target state of the environment, qualitative and quantitative characteristics of the intelligent agent, indicators of task completion, and expert evaluation. To address the challenge of selecting reward model components, effective methods for exploring the environment and the state-action space of agents are needed. The better the components of the reward model are chosen, the more efficient the reinforcement learning process will be. One or several parameters simultaneously can influence the agent’s reward. The use of the UR model and multi-agent reinforcement learning aims to enhance the usability and user experience of the mobile user interface. Therefore, the paper also examines the impact of standard MARL algorithms on the usability of the mobile user interface.

The paper is organized as follows. Section 2 provides a literature review. Section 3 describes the proposed model and the contribution of this paper. Section 4 discusses the experiment and the research environment for evaluating the proposed model’s effectiveness. Section 5 summarizes the obtained results and serves as the conclusion. At last, conclusions and future research issues are dealt with in Section 6.

2. Related Work

Research in the field of adaptive user interfaces and reinforcement learning (RL) algorithms has been conducted for several decades. Broadly, these studies can be divided into two major groups: improving usability and creating reward models.

2.1. Improving Usability

The user interface can be considered as a means of mapping user tasks to system tools [14]. In this context, context and adaptation are important characteristics of the user-system interaction that can be used to simplify the task of representation and thus enhance the interface. A system based on these features can adapt its actions according to the given context. For instance, the interface agent-assistant proposed in [21] learns to assist the user by observing their actions and imitating them, receiving feedback from the user when incorrect actions are taken.

Implicit learning of user preferences is implemented by observing their actions [22]. The agent analyzes user reactions to presented documents and extracts individual user profiles. Reinforcement learning is used to adapt weights in the user profile to represent user preferences optimally.

On the other hand, the motivation for creating adaptive human–machine systems that learn through trial and error is the automatic formation of a user-friendly human–machine dialogue [23]. Theoretical learning of such systems can be performed using human users or corpora of human–machine interactions. However, exploring the extensive space of possible dialogue states is impractical without using user action modeling tools.

The results of user research show that people use reward signals not only to receive feedback on past actions, but also to model future rewards by adjusting new actions [24]. This work demonstrates the importance of understanding the “human–teacher/agent–learner” system for developing algorithms that enhance the efficiency of intelligent agent learning.

Reinforcement learning approaches are used for mapping natural language instructions to sequences of executed actions [25]. It is assumed that the developer has access to a reward model that determines the quality of executed actions. The policy gradient algorithm is used to estimate the parameters of a linear model of action selection results, demonstrating that this approach can compete with supervised learning methods without requiring many annotated training examples. The approach is applied to interpret instructions of two types: troubleshooting guides for operating systems and tutorials for computer games.

Convenient user interfaces can be characterized by their ability to support logical reasoning, planning under uncertainty, short-term adaptation, and long-term experience-based learning [26]. The corresponding structure of the human–machine interaction for such interfaces can be based on the model of Partially Observable Markov Decision Processes (POMDPs). The advantages of this approach are demonstrated in the example of a simple iPhone application interface controlled by gestures. Moreover, the hypothesis that people use models like POMDP for action planning in uncertain conditions is considered. Users can directly manipulate system functions to express their interests, and reinforcement learning is used to model the user, allowing the system to solve famous in RL the exploration-exploitation dilemma [27]. This approach provides users with a more efficient way to manage their information search.

A conceptual reference structure for adapting the user interface has been defined, containing a set of adaptation properties [16]. This set of properties can be used to compare different methods and generate new ideas for adaptation. Although most software developers recognize the relevance and benefits of adaptation properties, they are not yet considered in the design phase. Perhaps the key task of the machine learning method is to propose the right adaptation at the right time and place to make it as valuable as possible for the end user. Otherwise, adaptation will be subject to limitations that may prevent the expected benefits if not addressed: the risk of mismatch, disruption of cognitive functions, lack of prediction, lack of explanations, lack of engagement, and risks regarding confidentiality.

A model-oriented approach to reinforcement learning is aimed at solving all the mentioned adaptation problems and expanding the scenarios of using adaptive interfaces [8]. The approach has been applied to adapt the layout of web page designs, placement of icons on the main screens of mobile devices, and reorganization of application menus. Depending on the application, the goal of adaptation may vary and include minimizing selection time, increasing the importance of elements, reducing cognitive load, increasing user engagement, or a combination of these goals. This approach can handle various types of adaptation, including the visual representation of graphic elements (position, size, color, etc.) or their behavior (number of elements, animation, etc.) [28]. The main idea of the approach is based on special models of the human–machine interaction, where it is necessary to accurately determine the impact of adaptation on selected functional goals.

2.2. Creating a Reward Model

A model-oriented optimization of user satisfaction during interaction is the main goal of an adaptive interface proposed by [29]. The reward model in this work is a simple sum of L1 and L2 distances between the vertices of the object (o) and the target (t):

$$r = -\sum_i (\|o_i - t_i\|_1 + \|o_i - t_i\|_2), \quad (1)$$

Thus, the reward model (1) measures the distance to the target in a user-defined interaction task.

After taking an action, the agent immediately receives a reward from the environment, set as a negative value of the expected time cost for each action [30]:

$$R(N) = R_d(N) + R_{\text{touch}}, \quad (2)$$

where $R(N)$ is the time cost of choosing a target from N candidates, $R_d(N)$ is the time cost of deciding which object to choose among N candidates, and R_{touch} is the time cost of the action of selecting a target. An empirical decision-making time model (in seconds) was used to evaluate $R_d(N) = 0.08 \log_2(N) + 0.24$, and the Fitts model was applied to estimate the touch time, representing an extended and refined form of Fitts' law [31].

With the help of Multi-Agent Reinforcement Learning (MARL) based on the popular Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm [32], it is possible to design adaptive systems by simulating reciprocal adaptation between a user and an intelligent tool [33]. In this case, the reward of the agents is defined by an indicator function, equal to -1 at each step of the agents and $+10$ if the agents together reach the target position. Thus, agents must collaborate to achieve the goal in the minimum number of steps. This multi-agent approach demonstrates that agents can learn realistic action policies, with cooperative agent actions being 30% more effective compared to a single-agent approach.

The reward model includes two weighted conditions, creating a compromise between task accuracy and task execution time: how much the current input observation x differs from the target state g and how much time is required for the action [15]. Therefore, the high-level action policy of the agent must consider how interface elements relate to the interaction task:

$$R_d = \alpha \varepsilon_{gd} - (1 - \alpha)(T_D + T_M) + \mathbb{1}_{\text{success}}, \quad (3)$$

where ε_{gd} is the difference between the input observation and the target state, α is the weighting coefficient, T_M is the predicted movement time, depending on the current position and endpoint parameters, T_D is the decision-making time, and $\mathbb{1}_{\text{success}}$ is an indicator function, equal to one if the task was successfully completed and zero otherwise.

The agent can receive an immediate reward from the environment once it takes an action [34]. The reward is determined as the negative value of the time cost for performing the action. The estimation of time costs was based on empirical data obtained during the study of menu selection by multiple users. At each step, the agent forms a vector of values $[sim, s_{\text{finger}}, s_{\text{focus}}, p_i]$, where sim is the similarity between the current interface element and the target element, s_{finger} is the position of the element for the mouse pointer, s_{focus} is the position of the element for the agent's selection, and p_i is the probability of the target element being in a specific position within the interface, based on memory positions. The range of s_{finger} and s_{focus} is normalized to $[0..1]$ for better learning performance. Each value in this range is specified by the distribution of menu item positions. The Deep Q-Network (DQN) algorithm [11] was used for agent training, which performs well in reinforcement learning tasks where the state space is continuous and the action space is discrete.

There is a separate research direction related to finding a pre-unknown reward model of the environment based on agent observation data. The model of internal symbolic rewards by [35] defines rewards as low-dimensional decision tree structures and uses them for standard reinforcement learning.

Table 1 presents the main methods of implementing the fundamental principles of the user interface adaptation, their models, and reinforcement learning algorithms. Reinforcement learning neural network algorithms demonstrate the key trends in modern deep multi-agent learning: independent learning, centralized training with decentralized execution, and leverage basic neural network architectures, such as fully connected, convolutional, recurrent, and actor-critic. The approach in this paper applies the RL algorithms—IQL (multi-agent modification of DQN), MADDPG (multi-agent modification of DDPG), and the Dec-POMDP model (multi-agent modification of POMDP)—for implementing an adaptive mobile user interface, which will be demonstrated in Sections 3 and 4.

Table 1. Reinforcement Learning Algorithms and Models in User Interface Adaptation Tasks.

RL Algorithm	Model	Author, Year, Reference
Memory-based learning	Closest distance	Maes et al., 1993 [21]
Q-Learning	MDP	Seo et al., 2000 [22]
Model-based Q-Learning	MDP	Schatzmann et al., 2006 [23]
Interactive Q-Learning	MDP	Thomaz et al., 2006 [24]
Policy Gradient	MDP	Branavan et al., 2009 [25]
Iterative optimization	POMDP	Young, 2010 [26]
Q-Learning	MDP	Glowacka et al., 2013 [27]
DQN	MDP	Mnih et al., 2015 [11]
DQN	MDP	Littman, 2015 [28]
DDPG	MDP	Debard et al., 2020 [29]
Maxmin DQN	POMDP	Sheikh et al., 2020 [35]
MCTS	MDP	Todi et al., 2021 [8]
DQN	MDP	Li et al., 2022 [30]
PPO	POMDP	Langerak et al., 2022 [15]
MADDPG	POMDP	Gupta et al., 2023 [33]
DQN	MDP	Li et al., 2023 [34]

MDP—Markov Decision Processes; POMDP—Partially Observable Markov Decision Process; DQN—Deep Q-Network; MADDPG—Multi-Agent Deep Deterministic Policy Gradient; MCTS—Monte Carlo Tree Search; PPO—Proximal Policy Optimization.

3. Method

3.1. Decentralized POMDP Model

The uncertainty associated with the inability of a single agent to obtain the exact state of the environment complicated the application of the classical Markov decision process model for reinforcement learning. To overcome the uncertainty of the environment's state, a mathematical model called the Partially Observable Markov Decision Process (POMDP) was developed, which, in the multi-agent case, is extended into a Decentralized POMDP (Dec-POMDP) model by introducing joint actions and joint observations of agents [36].

The Dec-POMDP model consists of a tuple: $(N, S, \mathbb{A}, T, R, \mathbb{O}, O, h, b^0)$, where N is the set of agents, S is the set of states, \mathbb{A} is the set of joint actions, T is the transition function, R is the reward function, \mathbb{O} is the set of joint observations, O is the observation function, h is the task's time horizon, and b^0 is the initial state distribution. The set of joint actions \mathbb{A} consists of $\mathbb{A} = \times_{i \in n} A_i$, where A_i is the set of actions of agent i . At each time t , agent i performs some action a_i , which leads to the formation of the joint action of the multi-agent system $\check{a} = \langle a_1, a_2, \dots, a_n \rangle$. The result of the joint action \check{a} effect on the environment is determined by the transition function T as the probability $T(s'|s, a)$ to transition from state s to state s' . The set of joint observations \mathbb{O} consists of $\mathbb{O} = \times_{i \in n} O_i$, where O_i is the set of observations available to agent i . At each time t , agent i receives some observation o_i , which forms the joint observation $o = \langle o_1, o_2, \dots, o_n \rangle$. The observation function O determines the probability of agents perceiving the environment's state as a joint observation $O(o|s', a)$.

In the Dec-POMDP model, the set of agents learns to optimize their actions by maximizing the expected cumulative reward $R_t = \sum_{t=0}^{h-1} \gamma^t r_t(o_t, a_t)$ with a discount parameter

$\gamma \in [0, 1]$ over some time horizon h , considering the initial state distribution of the environment $b^0 : S \rightarrow 0, 1$ at time $t = 0$ (Figure 1).

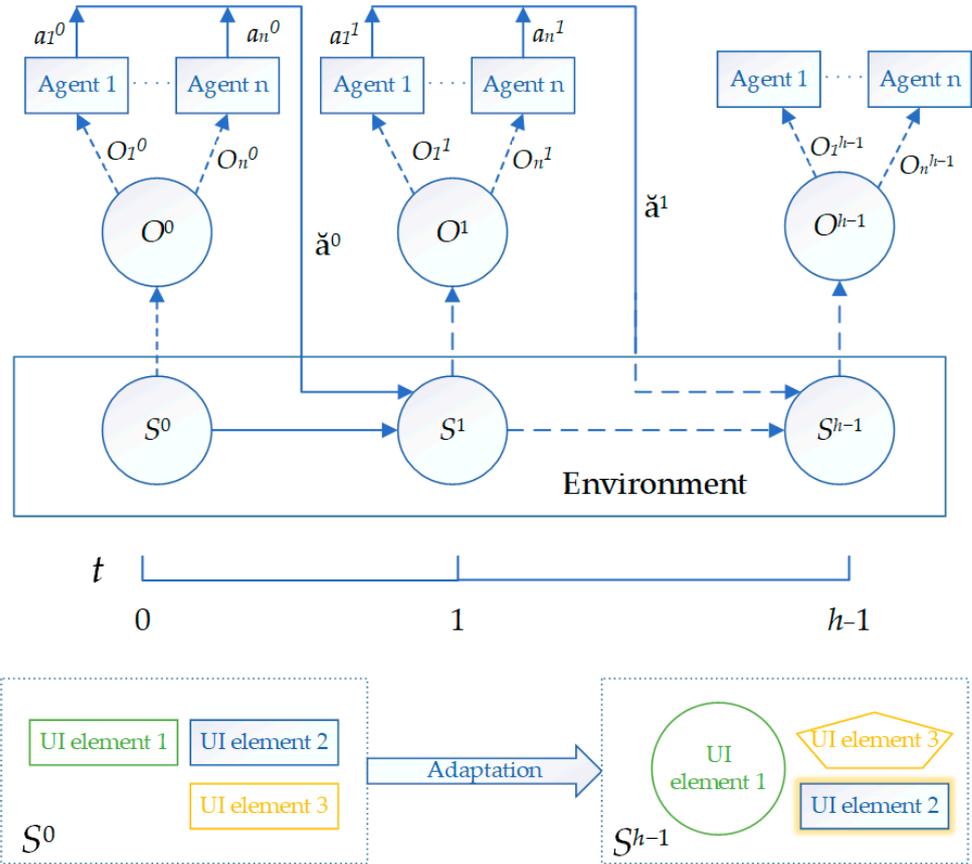


Figure 1. Dynamics of the Dec-POMDP model operation in a multi-agent mobile user interface.

3.2. Usability Reward Model

To solve the task of selecting reward model components, efficient methods for exploring the environment and the agents’ “state-action” space are needed. Additionally, to improve the accuracy and reliability of the reward model, a combination of parameters is required, such as the target state of the environment, qualitative and quantitative characteristics of the agent, usability metrics of the interface, and indicators of task completion. In the UR model, the reward of agent i is determined for each parameter k at time $t \neq 0$ by the formula:

$$r_{ik} = 1 - \beta \left| s_k^{(target)} - s_k^{(current)} \right|, \tag{4}$$

where $s_k^{(target)}$ is the target value of parameter k , characterizing the state of agent i and the environment, $s_k^{(current)}$ is the current value of the parameter, β is the coefficient of parameter k . The values of s_k are normalized in the range from 0 to 1.

The overall reward of agent i in the UR model consists of the sum of rewards across all k parameters, the sum of rewards across all u usability metrics, and the reward for task completion by the agent:

$$R_i^{UR} = \sum_k r_{ik} + \sum_u r_{iu} + \gamma \mathbb{1}_{success_i}, \tag{5}$$

where r_{ik} is the reward of agent i for each parameter k from (4), r_{iu} is the reward of agent i for each metric u from Table 2, $\mathbb{1}_{success_i}$ is the indicator function equal to 1 if the task of agent i is successfully completed (activated). The result of the $\mathbb{1}_{success_i}$ function is discounted by the coefficient $\gamma \in [0, 1]$.

Table 2. Measurable usability metrics.

Name of Usability Metric (Reference)	Definition
Structured Text Entry [37]	$STE = \frac{1}{n} \sum_i^n Structured_Text_Entry()$ <p>where <i>Structured_Text_Entry()</i> returns 1 if the input element displays a mask, 0 otherwise; <i>n</i>—the total number of input elements that accept data in an exact format.</p>
Essential Efficiency [38]	$EE = 100 \cdot \frac{S_{essential}}{S_{enacted}}$ <p>where <i>S_{essential}</i> is the number of user steps in the description of the primary (best) use case, and <i>S_{enacted}</i> is the number of steps required to perform the use case with the current user interface design.</p>
Built in Icon [37]	$BI = \frac{1}{n} \sum_i^n Built_in_Icon()$ <p>where <i>Built_in_Icon()</i> returns 1 if the action element displays a system icon, 0 otherwise; <i>n</i>—the total number of action elements in the interface.</p>
Density Measure [37]	$DM = 1 - \frac{1}{a_{frame}} \sum_i^n a_i$ <p>where <i>a_i</i> and <i>a_{frame}</i> represent the area of object <i>i</i> and the area of the frame, respectively, and <i>n</i> is the number of objects in the frame.</p>
Layout Appropriateness [38]	$LA = 100 \cdot \frac{C_{optimal}}{C_{designed}}$ $C = \sum_{\forall i \neq j} P_{i,j} \cdot D_{i,j}$ <p>where <i>P_{i,j}</i> is the transition frequency between visual components <i>i</i> and <i>j</i>, <i>D_{i,j}</i> is the distance between visual components <i>i</i> and <i>j</i>.</p>
Default Value [37]	$DV = \frac{1}{n} \sum_{i=1}^n a_i$ <p>where <i>a_i</i> is the input element with a default value, <i>n</i> is the total number of input elements.</p>
Task Concordance [38]	$TC = 100 \cdot \frac{2 \cdot D}{N(N-1)}$ <p>where <i>N</i> is the number of tasks to be ranked, <i>D</i> is the inconsistency index, i.e., the number of tasks pairs whose difficulties are arranged correctly minus those pairs whose difficulties are arranged incorrectly.</p>
Target Element Size [37]	$TeS = \frac{1}{n} \sum_{i=1}^n a_i$ <p>where <i>a_i</i> returns 1 if the area of object <i>i</i> is greater than or equal to 44 pt × 44 pt (or 88 pt × 88 pt), otherwise 0, and <i>n</i> is the number of interactive objects in the interface.</p>
Text Size [37]	$TxS = \frac{1}{n} \sum_{i=1}^n FontSize_i$ <p>where <i>FontSize_i</i> returns 1 if the font size for text input <i>i</i> is greater than or equal to 16 px, otherwise 0, and <i>n</i> is the number of text inputs in the interface.</p>
Task Visibility [38]	$TV = 100 \cdot \frac{V_i}{S_{total}}, \forall i$ <p>where <i>S_{total}</i> is the total number of implemented steps to complete the use case, <i>V_i</i> is the visibility of the implemented step <i>i</i>, ranging from 0 to 1.</p>
Balance [37]	$BL = 1 - \frac{ BL_{vert} + BL_{hor} }{2}$ <p>where <i>BL_{vert}</i> is the vertical balance, and <i>BL_{hor}</i> is the horizontal balance.</p> $BL_{hor} = \frac{W_L - W_R}{\max(W_L , W_R)}$ $BL_{vert} = \frac{W_T - W_B}{\max(W_T , W_B)}$ <p>where <i>L</i>, <i>R</i>, <i>T</i>, and <i>B</i> refer to the left, right, top, and bottom edges, respectively. <i>W_j</i> is the weight of the <i>j</i>-th side of the interface (left, right, top, and bottom).</p>

Table 2. Cont.

Name of Usability Metric (Reference)	Definition
Horizontal (BH) or Vertical Balance (BV) [38]	$Bx = 200 \cdot \frac{W_1}{W_1 + W_2}$ where W_1 is the weight of the first side, W_2 is the weight of the second side. The weight of a side (W_i) = the number of pixels used \times the distance of the side from the center. The center = halfway between the left edge of the leftmost visual element and the right edge of the rightmost element.
Element Readability [8]	$TR(i_l) = \frac{\delta}{1 + B(i_l)}$ where $B(i_l)$ is the activation of element i at point l , and δ is the constant for careful examination of the element in the absence of activation.
Element Selection [8]	$TP(i_l) = a_p + b_p \cdot \log(1 + i_l)$ where a_p and b_p are the Fitts' law constants for estimating the time to select an element (for example, $a_p = 10.3$ and $b_p = 4.8$), for the target element i at point l .
Local Search [8]	$TL(i_l) = T_c + \delta \cdot N_{local} + T_{trail}$ where T_c is a constant penalty for unexpectedness (element found where it was not expected), δ is the constant cost of careful examination of the element, N_{local} is the number of nearby elements being checked, T_{trail} is the constant time for the cursor to follow the gaze of the eye.

The average reward value of the agent i for each parameter k without considering the indicator function $\mathbb{1}_{\text{success}_i}$ can also be used in determining the quality of agent actions.

$$r_{i\text{mean}} = \langle r_{ik} \rangle, \quad (6)$$

In Section 4, we will demonstrate how the reward is computed using expressions (4)–(6) using the example of a multi-agent environment of a mobile kitchen user interface. Measurable usability metrics will allow expanding the reward model components to enhance the quality of the adaptive interface. Table 2 presents popular metrics that can be used in the Usability Reward model.

The usability of the interface is automatically calculated by the environment, where a reinforcing reward signal is generated based on the presented metrics (Table 2), and it is considered in the reward model of the multi-agent environment after each action performed by the agents. The values of the usability metrics r_{iu} , according to the expressions described above, are in the range from 0 to 1, where approaching $r_{iu} \rightarrow 0$ indicates low interface quality and $r_{iu} \rightarrow 1$ indicates high quality.

3.3. IQL Algorithm

Thus, the adaptive user interface includes the application of the Dec-POMDP model, defining the agent's interaction process with the environment, the UR model, generating rewards, and the Independent Q-Learning (IQL) algorithm, performing multi-agent learning. The IQL algorithm using a fully connected neural network is based on the classical DQN algorithm [39]. Three main innovations borrowed by IQL in deep Q-learning are as follows: the use of deep neural networks, the utilization of a replay buffer, and a target neural network.

The IQL algorithm implements the idea of independent learning by embedding an identical neural network architecture into each individual agent, considering the absence of explicit communication between agents. The main and target neural networks are initialized with random weights $\theta \leftarrow \xi$ and $\theta' \leftarrow \xi$ (step 1). The replay buffer E_b is initialized as empty, with a capacity B_l set to 10,000. The batch size B_s from the replay buffer is set to 32. The weight copying period C_s for the main and target neural networks is chosen to be 100 episodes. To explore the agent's action space in the IQL algorithm, the concept of ε -greedy action selection is applied (step 5). The value of ε linearly decreases over time from 1.0 to 0.1. The agent's selection of a possible action a is performed according to the maximum Q -value of the neural network's nonlinear approximator θ for state s ,

neural network weights θ , and considering the ε parameter. The IQL algorithm for agent i , where α is the learning rate; γ is the discount factor for future rewards; ε is the exploration-exploitation parameter; N_e is the number of episodes, is presented Algorithm 1 below.

Algorithm 1 Deep Independent Learning for Agent i Using Fully Connected Neural Network.

1. Initialize:

$$\alpha \in (0, 1]; \gamma \in [0, 1]; \varepsilon \in [0.1, 1]; N_e \in [1, N]; E_b \leftarrow 0; B_l \leftarrow 10,000; B_s \leftarrow 32; C_s \leftarrow 100; \\ i \leftarrow 1, \dots, n; a \in A_i; j \leftarrow 1, \dots, B_s; K \leftarrow |A_i|; \theta \leftarrow \zeta, \theta' \leftarrow \zeta, \forall s \in S.$$

2. Loop.

3. Get the state s .

4. Determine the possible actions $a \in A_i$ in state s .

5. Pass the state s into the main neural network and select a possible action a according to the maximum value of $Q_1(s, a; \theta), Q_2(s, a; \theta), \dots, Q_K(s, a; \theta)$, considering the parameter ε .

6. Perform the possible action a .

7. Obtain the reward R_i^{UR} from the UR model and the next state s' .

8. Save (s, a, R_i^{UR}, s') in the replay buffer E_b . If the capacity B_l is exceeded, replace the value according to the FIFO queue principle.

9. Randomly select a minibatch (s, a, R_i^{UR}, s') of size B_s from the replay buffer E_b .

10. For each tuple (s, a, R_i^{UR}, s') in the minibatch, calculate $Q^j(s', a'; \theta')$, by passing s' to the target neural network.

11. Calculate an intermediate variable to assess the optimality of the agent's actions:

$$y^j \leftarrow \begin{cases} R_i^{UR}, & \text{if } t = T \\ R_i^{UR} + \gamma \max_{a'} Q^j(s', a'; \theta'), & \text{if } t \neq T. \end{cases}$$

12. For each j tuple (s, a, R_i^{UR}, s') in the minibatch, calculate $Q^j(s, a; \theta)$, by passing s to the main neural network.

13. Calculate the loss function:

$$L^j(\theta) \leftarrow \frac{1}{K} \sum_{p=1}^K (y_p^j - Q_p^j(s, a; \theta))^2.$$

14. Update the weights θ of the main neural network using $L^j(\theta)$ and the learning rate α .

15. Replace the weights θ' of the target neural network with the weights θ of the main neural network if the next period C_s has ended.

The main advantages of the IQL algorithm are as follows: simplicity of implementation, speed of operation, decentralized architecture, and robustness in achieving good results when applied to various practical tasks. This algorithm is iterative, involving decisions about adaptations (changes in the interface element's state) based on differences between the original and target user interfaces formed from data obtained from human-machine interaction processes and pre-formed usability context.

4. Experiment

4.1. Multi-Agent Environment

For conducting comparative experiments on multi-agent reinforcement learning was developed a "sandbox" platform of a mobile user interface. The platform is based on the popular Kivy software library v2.0.0, which provides rapid application development capabilities for applications equipped with new user interfaces, such as multi-touch applications [40]. Based on multithreading, decentralized control over a multitude of interface elements-widgets deployable in popular mobile ecosystems like Android is ensured. Each widget element is controlled by an independent learning agent, having access only to local environment observations.

Figure 2 shows the graphical interface of the developed multi-agent environment of a mobile kitchen user interface. The investigated environment addresses the adaptation of the mobile user interface in solving user tasks—cooking dishes.



Figure 2. Multi-agent environment of a mobile kitchen user interface.

In the interface of the multi-agent environment, usability metrics (Figure 3) are implemented based on expressions from Table 2: density metric (DM), target element size (TeS), balance (BL), element readability (TR), element selection (TP), local search (TL), layout appropriateness (LA), task visibility (TV), horizontal balance (BH), and vertical balance (BV). Figure 3 shows the developer interface for selecting UR model components and configuring the target user interface vector.

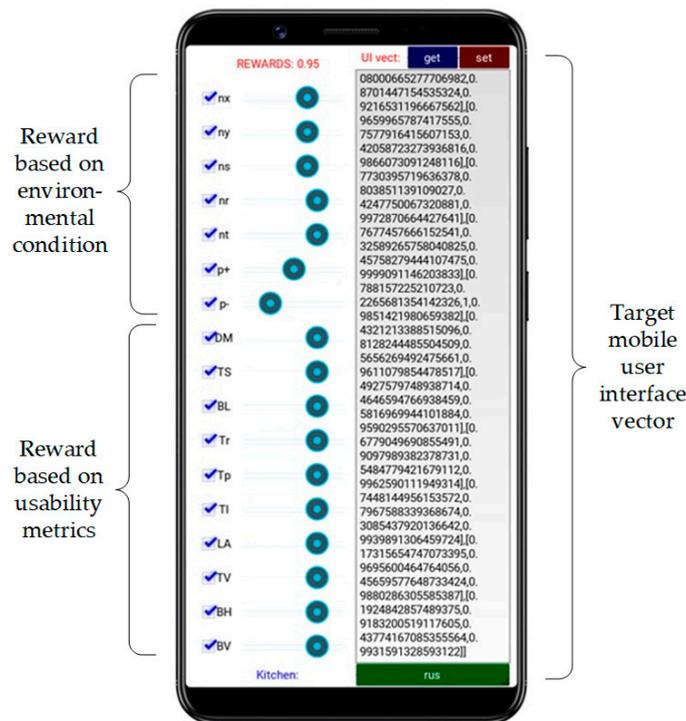


Figure 3. Developer interface for selecting UR model components—usability metrics for reward computation (Table 2): DM, TeS, BL, TR, TP, TL, LA, TV, BH, BV.

Due to the lack of required input data for the solved task in the developed environment, the following metrics were not utilized (Table 2): structured text entry (STE), essential efficiency (EE), built-in icons (BIs), default value (DV), text size (TxS), and task consistency (TC). For example, for the “Essential Efficiency (EE)” metric, the number of user steps in the description of the main (best) use case is not known in advance, and for the “Default Value (DV)” metric, the input elements with default values are unknown.

4.2. Use Cases

The developed multi-agent environment includes the use of interface agents representing basic ingredients from three different cuisines: European, Slavic, and Asian. In such a reinforcement learning task, the state space and the action space are discrete.

The actions of the agents include the following operations: moving the element in four different directions, rotations, and resizing (Figure 4). The state vector of each interface element (agent) includes the identifier, number of clicks, X position, Y position, size, and rotation.

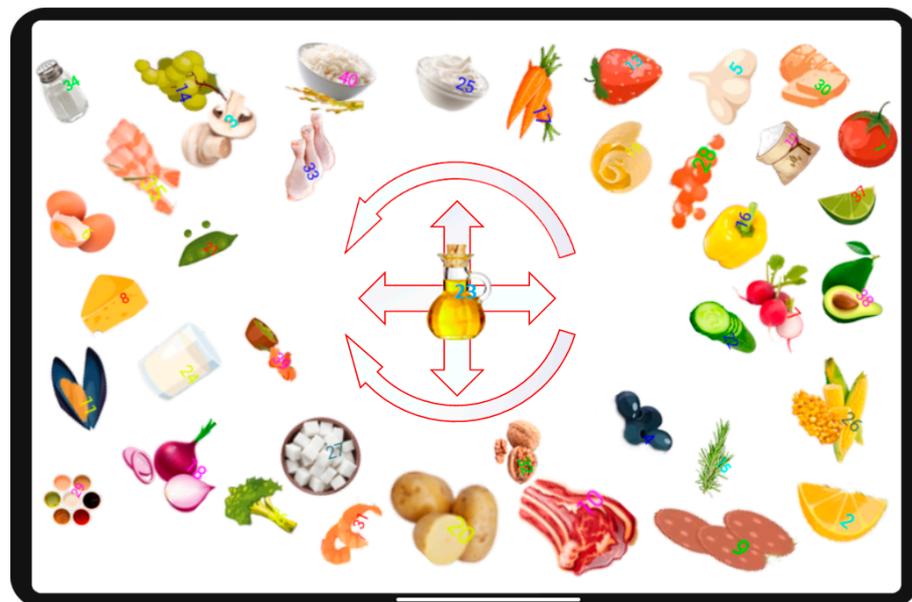


Figure 4. Actions of interface agents: move left, move right, move down, move up, rotate, resize.

This work conducts an experiment to find the target state of the kitchen interface using three cuisine categories European, Slavic, and Asian and the user—chef, preparing dish ingredients. Adaptation involves placing dish elements (moving, resizing) so that the user can quickly select ingredients for dishes of a particular cuisine.

The target user interface is a user-friendly interface for performing a specific user task, such as cooking dishes from a particular cuisine. In such an interface, all elements are given specific positions and sizes, and some may be absent altogether. However, the target interface is unknown in advance and can only be determined through online adaptations based on human–machine interaction and multi-agent reinforcement learning. The results of the usability experiment to find the target user interface are presented in Figure 5. The experiment was conducted with students comprising 50 participants (33 masculine, 17 feminine, and 0 others) aged 19 to 23 (mean 22) who were residents of Eastern Europe with varying educational backgrounds and were opportunistically recruited. All participants reported frequent desktop or mobile and web usage.

The source code of the developed multi-agent environment, based on the Dec-POMDP model, which defines the agent’s interaction with the environment, the UR model, which generates rewards, and the IQL and MADDPG algorithms for multi-agent learning, is available at: <https://github.com/dimi3o/MobAdaptUi> (accessed on 23 March 2024).



Figure 5. Target Mobile User Interface—Kitchen: European (a), Slavic (b), Asian (c).

4.3. Reward Computation

For the proposed environment, the state space of each interface agent is defined by the parameters as follows: horizontal s'_x and vertical position s'_y , scale s'_s and tilt s'_a of the element. The parameter s'_t , representing the number of user touches, characterizes the agent's task completion. Thus, the state s'_k of the agent in expression (4) is defined by a tuple of values (s'_x, s'_y, s'_s, s'_a) , which is determined for the current $s'_k^{(current)}$ and target $s'_k^{(target)}$ interface states. Therefore, for calculating the total reward in expression (5), there will be $k = 4$ local rewards r_{ik} , each of which returns a value in the range from 0 to 1. If the reward r_{ik} assigns a value close to 0, then this parameter k significantly differs from the target, and if the value is close to 1, it is close to the target.

The indicator function $\mathbb{1}_{\text{success}i}$ in expression (5) equals 1 if task i of the agent is completed. Thus, the parameter s'_t , the number of users touches, defines the value of $\mathbb{1}_{\text{success}i}$ when the element is activated by the user. For example, if the user clicks on the element, the agent-element is rewarded 0.1 for each click. The maximum value of $\mathbb{1}_{\text{success}i}$ is 1 if the user makes enough clicks (activation).

If an element is positioned as in the target interface (Figure 5) and activated, the average arithmetic reward of the corresponding agent-element approaches 1. Therefore, for $k = 4$ parameters and local rewards r_{ik} , parameter s'_t , and the indicator function $\mathbb{1}_{\text{success}i}$ the maximum total reward of agent i resulting from performing one action (5): $R_i \leq 5$.

The graph showing the average value of the reward r_{imean} obtained from the experiment, as described in Equation (6), is presented in Figure 6. This graph illustrates the quality of agent actions without considering user clicks activation. Interface adaptation occurred over a time interval of 140 s, encompassing 1000 adaptations for each agent. To alter the interface state, agent actions were executed randomly. The decreasing trend in the reward function indicates a deterioration in interface usability in the absence of a defined agent action policy.

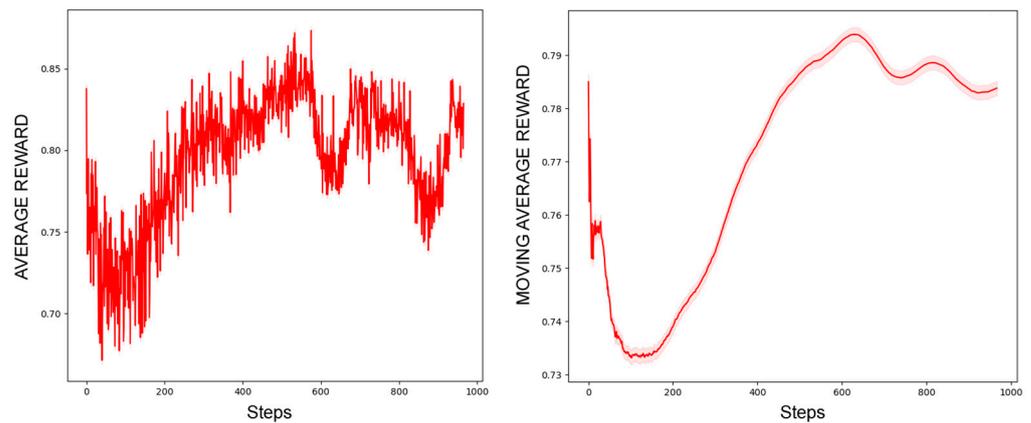


Figure 6. The average reward value of agent i in the UR model over the course of an episode (1000 adaptations).

Under similar conditions, the expected cumulative average reward of all agents

$R_{tmean} = \sum_{t=0}^{h-1} \gamma^t r_{imean}$ with a discount factor of $\gamma = 0.99$ over the course of an episode (14 s, 2000 adaptations) reaches a plateau of limiting values, indicating the completion of the user interface usability improvement phase, is presented in Figure 7.

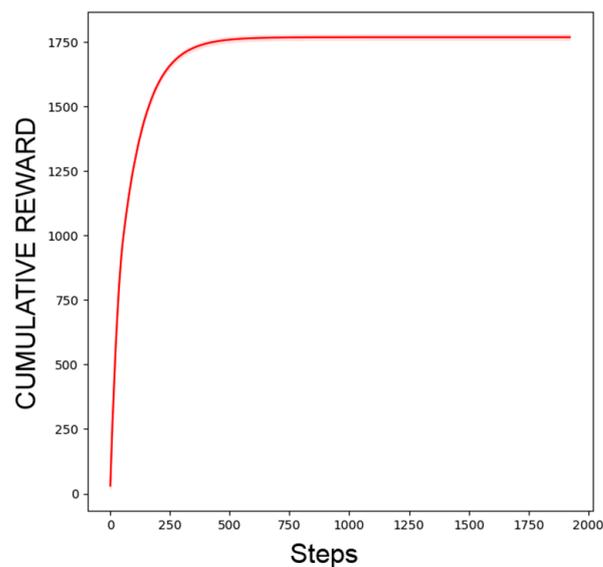


Figure 7. Expected cumulative average reward of the set of interface element agents over the course of an episode (2000 adaptations).

4.4. Experimental Results and Analysis

Figures 8–10 show the results of Mobile User Interface (MUI) adaptation experiments during training using the multi-agent reinforcement learning algorithms IQL and MAD-DPG. The training episodes were set to a horizon of 200 environment adaptations. The results are aggregated over several dozen episodes (steps) to calculate the moving average loss during training (mean loss), as well as the local (reward) and cumulative rewards (total reward) for the agents.

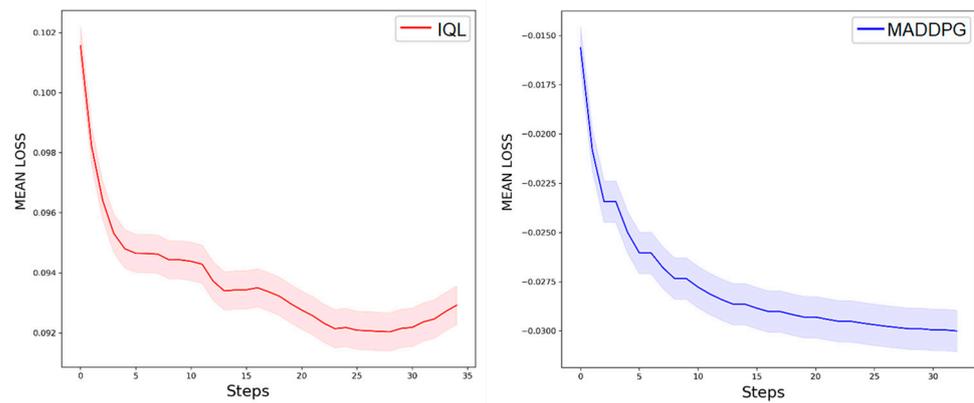


Figure 8. Moving average loss during the training process for the IQL and MADDPG algorithms.

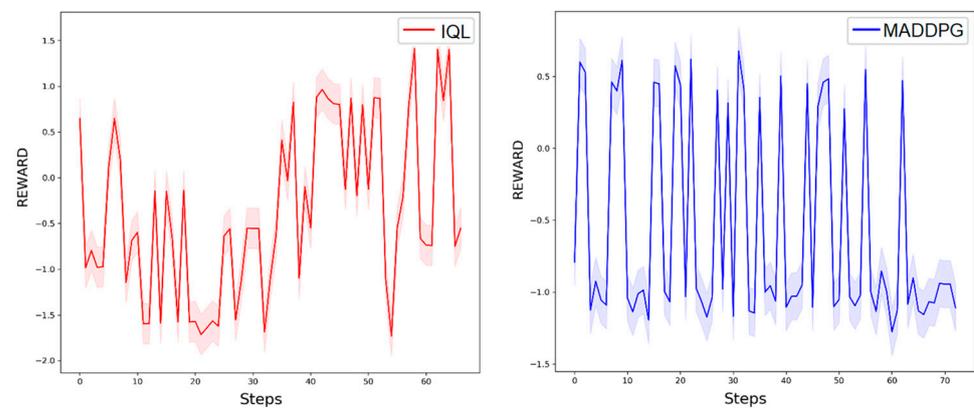


Figure 9. Local reward based on the UR model in multi-agent learning for the IQL and MADDPG algorithms.

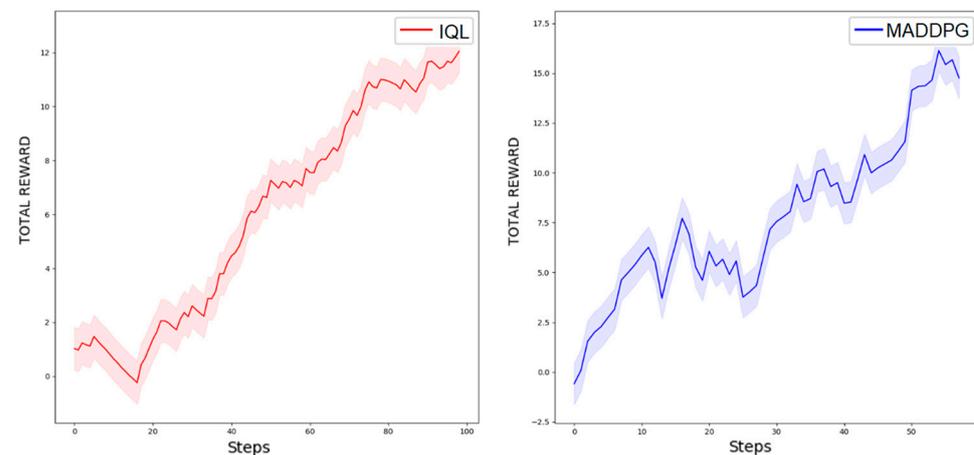


Figure 10. Cumulative reward based on the UR model in multi-agent learning for the IQL and MADDPG algorithms.

Usability testing of the adapted interfaces was conducted using the Questionnaire for User Interface Satisfaction (QUIS) 7.0 methodology [41,42], and key usability characteristics have been highlighted in Figure 11. Thus, an analysis of the quality and usability of the mobile kitchen user interface obtained through adaptation based on the IQL and MADDPG algorithms is performed, along with their comparison in terms of usability criteria [42,43].

	Adaptation by IQL		Adaptation by MADDPG
			
Usability criteria	IQL	MADDPG	
Effectiveness	+	–	
Error management	–	–	
Significance of codes	–	–	
Consistency	+	–	
Discriminability	+	+	
Comprehensibility	–	–	
Detectability	+	+	
Guidance	+	–	
Adaptability	+	+	
Consistency	–	–	
Satisfaction	+	+	

Figure 11. Comparison of the quality of adaptation of the MUI in usability criteria.

The resulting interface based on adaptation using the IQL algorithm, according to users, meets the “Effectiveness” criterion. This criterion signifies the interface’s ability to enable users to accomplish tasks with minimal effort and time. In general, the higher the efficiency of the interface, the better users can achieve their goals and complete tasks with fewer time and effort expenditures. However, the adaptation result based on the IQL and MADDPG algorithms was perceived by users as unclear, meaning it does not meet the Comprehensibility criterion. This criterion measures the level of clarity and comprehensibility of the interface based on labels, instructions, and other interface elements, as well as the ease of understanding and using available functions and features. Importantly, in the survey results, users considered the obtained interface to meet the “Adaptability” criterion. That is, according to users, the obtained interface is adapted for use in various contexts: solving a task (preparing dishes of a certain cuisine), different devices, and screen resolutions.

4.5. Model Validation

The proposed approach based on the UR model was evaluated through empirical assessment of the time taken to search for and select user interface elements of interest, compared to two interfaces: a typical static interface (Static) and a frequency-adaptive interface (Frequency) [44]. The static interface represents a grid world where all elements are stationary and positioned randomly. In the frequency-adaptive interface, elements are moved depending on the selected priority position on the screen and their frequency of use and are also arranged in a grid. In the UR model approach, a set of RL agents interact with the environment, learning through trial and error to find the optimal interface state and adjusting the position and size of elements in any way.

To compare these approaches, an experiment was conducted in which participants performed the task of selecting an element under three conditions (Static, Frequency, and UR model). Fifty participants aged 18 to 24 (average age 21) with varying levels of education were involved. All participants reported consistent active use of mobile devices. Interface layouts with 40 icons of dish ingredients were randomly generated for the experiment. The icons were evenly selected from common categories: European, Slavic, and Asian cuisines. An interface was created for each of the three conditions for each participant. The name of the target element was displayed in a tooltip at the top of the application window. Participants had to select the target element correctly to complete the trial. Clicking on the target element hid the menu and provided a short pause. The selection time was recorded automatically.

One-way analysis of variance of the search and selection time (in milliseconds) as the dependent variable, with the conditions Static, Frequency, and UR model as the fixed independent variable, is presented in Figure 12. Vertical bars in the figure indicate 95% confidence intervals. The condition had a statistically significant effect on the selection time with large means for Static = 1285 ms, Frequency = 1273 ms, and the UR model = 1168 ms. Post-hoc Tukey-Kramer test showed that the UR model (1168 ms) was significantly faster (shorter search and selection time) than both Frequency (1273 ms) and Static (1285 ms); the difference between Static and Frequency was not statistically significant. Thus, the difference in search and selection time between the UR model and Frequency is 105 ms (i.e., the Frequency approach is approximately 9% slower than the UR model).

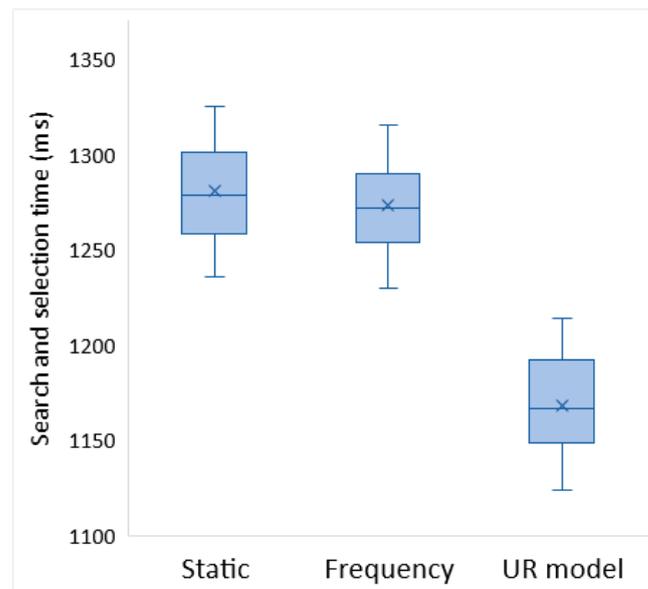


Figure 12. Difference in search and selection time for elements between the UR model approach in the current study and other interfaces, Static and Frequency.

5. Discussion

5.1. Limitations

The main foundation for the approach considered in the paper is multi-agent reinforcement learning, which offers its advantages and imposes its limitations [45].

Overall, the multi-agent approach in constructing a system in a nondeterministic and dynamic environment provides the following advantages:

- **Robustness:** The ability of a multi-agent system to continue functioning in the event of failures in individual agents' operations.
- **Efficiency:** Parallel data processing in a multi-agent system by autonomous agents accelerates the system's operation.
- **Adaptability:** A decentralized multi-agent system is capable of dynamically changing its behavior in response to changes in the environment.
- **Encapsulation:** A multi-agent system is modular by nature, allowing flexible changes to its structure, abstraction of data, and protection of internal implementations.
- **Scalability:** In information perception and decision-making, a multi-agent system has no limitations on centralized data control, which enables better handling of increased workloads with added resources.

However, applying machine learning methods to multi-agent systems also leads to new challenges.

Firstly, multi-agent learning suffers from the so-called "curse of dimensionality," caused by the exponential growth of the state-action space with an increase in the number of agents. For example, in the approach described in this paper, single-agent reinforcement

learning with 40 agents and eight possible actions, 320 Q-values would need to be stored for each state of the environment. However, in multi-agent learning with joint actions, it would be necessary to consider already 1.329×10^{36} Q-values.

Secondly, many theoretical guarantees that existed and were formally proven in single-agent learning no longer apply. For instance, the classical Q-learning algorithm guarantees convergence to an optimal strategy for a finite Markov Decision Process. But in multi-agent learning, in the considered case of a mobile user interface with the Dec-POMDP model, agents have to settle for not an optimal but some averaged equilibrium achieved in interactions with each other and the environment.

Thirdly, the exploration–exploitation problem becomes much more complex in multi-agent learning, as the transition of one agent from exploration to exploitation or vice versa implicitly affects the strategies of all other agents. For example, in the mobile user interface being considered, this means that the interface elements will be inconveniently positioned, usability criteria will not be met, and elements will move only in one direction and be pressed against the screen edges, which is very inconvenient for the user to interact with.

Finally, the fourth challenge is related to coordination. The effect of each agent’s action depends on the actions/inactions of other agents. Balancing between finding effective joint actions of agents and searching for optimal strategies is a constant challenge in coordination. Although this problem is more typical for cooperative multi-agent learning, the lack of coordination negatively affects independent learning agents as well. For example, due to the problem of multi-agent coordination, interface elements and agents from one type of cuisine may not immediately adapt to Slavic, European, and Asian cuisines.

5.2. Future Directions

Multi-agent systems, as a distinct discipline in computer science and computer engineering, began to develop at the end of the last century. Over the past twenty years, interest in this area has steadily increased [46], leading today to over half a million search results in the “Google Scholar” system.

On the one hand, this growth is associated with the belief that multi-agent systems are one of the promising paradigms for software development. The shift of focus from developing applications for personal workstations to developing distributed systems that interact remotely with each other has led to the complexity of these systems and their acquisition of properties characteristic of multi-agent systems.

On the other hand, the key property of an agent operating in an unknown environment is the ability to learn based on the experience gained from interacting with this environment or with other agents. The evident successes in recent years of deep single-agent learning in recognition and gaming tasks motivate researchers and developers to extend the potential of these methods to multi-agent learning.

Multi-agent learning today is a fundamental component of multi-agent systems. From a scientific perspective, researching the interaction between a multitude of simultaneously learning agents yields new knowledge that can be used in analyzing social structures of society, economically competitive environments, and control theory. From an engineering perspective, a multitude of simultaneous learning agents provides a conceptually new way of developing communication systems, robotics, transportation systems, human–machine interfaces, and software.

Currently, the main trend in AI is Large Language Models (LLMs). For example, recent works have successfully utilized LLM capabilities to gather abstract knowledge about global physics for decision-making problem-solving [47,48]. Applying LLM in the approach of the current work is challenging because LLMs weigh tens of gigabytes and cannot be loaded onto a mobile device for autonomous use. Additionally, we require online learning; thus, our approach can function as grounding RL [47]. Thus, the approach in the work encompasses this trend of LLM with RL.

The experiments were conducted on the Android mobile operating system. However, it is possible to port the entire experimental setup to other mobile operating systems. In the

development of the mobile user interface, including the functions of various applications, inter-process communication technology is commonly used for data exchange between threads. Similar approaches enable applications to communicate with each other and with the operating system through data requests, function calls, and responses to signals. This opens broader possibilities for the use of the adaptive mechanism of multi-agent reinforcement learning in a full technology stack of mobile interfaces.

6. Conclusions

The usability reward model in multi-agent reinforcement learning for the mobile user interface is explored in this study. It consists of parameters of the target state of the environment, qualitative and quantitative characteristics of the agent, and a task completion indicator. Classic Dec-POMDP model, the independent and actor-critic deep learning algorithms were employed for multi-agent learning. The proposed UR model allows for explicit reward determination in the mobile digital environment, reducing the time for selecting user interface elements.

An original multi-agent sandbox environment is developed, enabling the collection of high-quality user data autonomously for various human–machine interaction tasks. It involves the ability to create interface elements as agents adapting to usage scenarios, simultaneous training of multiple agents based on the multi-agent reinforcement learning approach, and real-time interface adaptations. A key feature of this work is the attempt to investigate the impact of multi-agent reinforcement learning (MARL) algorithms on the usability of mobile user interfaces. Usability tests have shown that the IQL and MADDPG algorithms have their own strengths and weaknesses in terms of a positive user experience. However, the IQL algorithm has some advantages in terms of usability criteria.

Computational experiments in the proposed environment show that a user-friendly interface (based on the criterion of minimal element selection time) can be defined through online adaptations using human–machine interaction and multi-agent reinforcement learning. A comparison of the search and selection times between the UR model and two other interface types, Static and Frequency, reveals a difference of 105 ms, with the Frequency approach being approximately 9% slower than the UR model.

Finally, these results also indicate that RL remains one of the leaders in the field of tinyML and user interfaces, opening up new levels of usability for users on the one hand, and on the other hand, in the technical aspect, laying the foundation for the full integration of LLM models into mobile agents alongside RL.

Author Contributions: Conceptualization, D.V. and A.A.; methodology, A.A.; software, D.V.; validation, D.V. and A.A.; formal analysis, D.V.; investigation, D.V.; resources, D.V.; data curation, D.V.; writing—original draft preparation, D.V.; writing—review and editing, D.V. and A.A.; visualization, D.V.; supervision, D.V. and A.A.; project administration, A.A.; funding acquisition, D.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Written informed consent was obtained from all subjects involved in this study.

Data Availability Statement: The data collected during the study are available upon request.

Acknowledgments: The work was carried out as part of the research work PRIOR/SN/NU/22/SP2/1.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UR	Usability Reward
MUI	Mobile User Interface
HCI	Human–Computer Interaction
RL	Reinforcement Learning
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Processes
POMDP	Partially Observable Markov Decision Process
DQN	Deep Q-Network
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MCTS	Monte Carlo Tree Search
PPO	Proximal Policy Optimization
IQL	Independent Q-Learning
LLM	Large Language Models
tinyML	Tiny Machine Learning

References

1. Tao, K.; Edmunds, P. Mobile APPs and Global Markets. *Theor. Econ. Lett.* **2018**, *8*, 1510–1524. [CrossRef]
2. Falloon, G. Mobile Devices and Apps as Scaffolds to Science Learning in the Primary Classroom. *J. Sci. Educ. Technol.* **2017**, *26*, 613–628. [CrossRef]
3. Lee, H. Mobile App Evolution: How the Ecosystem Has Changed. App Annie. Available online: <https://www.data.ai/en/insights/market-data/mobile-app-evolution-how-the-ecosystem-has-changed/> (accessed on 20 March 2024).
4. Ullah, I.; Boreli, R.; Kanhere, S.S. Privacy in targeted advertising on mobile devices: A survey. *Int. J. Inf. Secur.* **2022**, *22*, 647–678. [CrossRef] [PubMed]
5. Iqbal, M.W.; Ahmad, N.; Shahzad, S.K.; Feroz, I.; Mian, N.A. Towards adaptive user interfaces for mobile-phone in smart world. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 556–565. [CrossRef]
6. Zhou, J.; Tang, Z.; Zhao, M.; Ge, X.; Zhuang, F.; Zhou, M.; Xiong, H. Intelligent exploration for user interface modules of mobile app with collective learning. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Long Beach, CA, USA, 20 August 2020; KDD 20. Association for Computing Machinery: New York, NY, USA, 2020; pp. 3346–3355. [CrossRef]
7. Macías-Escrivá, F.D.; Haber, R.; del Toro, R.; Hernandez, V. Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Syst. Appl.* **2013**, *40*, 7267–7279. [CrossRef]
8. Todi, K.; Bailly, G.; Leiva, L.; Oulasvirta, A. Adapting user interfaces with model-based reinforcement learning. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; pp. 1–13. [CrossRef]
9. Gorban, A.N.; Pokidysheva, L.I.; Smirnova, E.V.; Tyukina, T.A. Law of the minimum paradoxes. *Bull. Math. Biol.* **2011**, *73*, 2013–2044. [CrossRef] [PubMed]
10. Gorban, A.N.; Smirnova, E.V.; Tyukina, T.A. General Laws of Adaptation to Environmental Factors: From Ecological Stress to Financial Crisis. *Math. Model. Nat. Phenom.* **2009**, *4*, 1–53. [CrossRef]
11. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
12. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
13. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef]
14. Croft, W.B. The role of context and adaptation in user interfaces. *Int. J. Man Mach. Stud.* **1984**, *21*, 283–292. [CrossRef]
15. Langerak, T.; Christen, S.; Albaba, M.; Gebhardt, C.; Hilliges, O. MARLUI: Multi-Agent Reinforcement Learning for Goal-Agnostic Adaptive UIs. *arXiv* **2022**, arXiv:2209.12660. [CrossRef]
16. Abrahão, S.; Insfran, E.; Sluÿters, A.; Vanderdonckt, J. Model-based intelligent user interface adaptation: Challenges and future directions. *Softw. Syst. Model.* **2021**, *20*, 1335–1349. [CrossRef]
17. Figueiredo, D.G. Learning from Interaction: User Interface Adaptation using Reinforcement Learning. *arXiv* **2023**, arXiv:2312.07216.
18. Singh, S.; Lewis, R.L.; Barto, A.G.; Sorg, J. Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective. *IEEE Trans. Auton. Ment. Dev.* **2010**, *2*, 70–82. [CrossRef]
19. Gaspar-Figueiredo, D.; Fernández-Diego, M.; Abrahao, S.; Insfran, E. A Comparative Study on Reward Models for UI Adap-tation with Reinforcement Learning. *Methods* **2023**, *13*, 14.
20. Razevicius, G.; Roudaut, A.; Karnik, A. HoberUI: An Exploration of Kinematic Structures as Interactive Input Devices. *Multimodal Technol. Interact.* **2024**, *8*, 13. [CrossRef]
21. Maes, P.; Kozierok, R. Learning interface agents. In Proceedings of the 11th National Conference on Artificial Intelligence, Washington, DC, USA, 11–15 July 1993; pp. 459–465.

22. Seo, Y.W.; Zhang, B.T. A reinforcement learning agent for personalized information filtering. In Proceedings of the 5th International Conference on Intelligent User Interfaces; IUI '00, New Orleans, LA, USA, 9–12 January 2000; Association for Computing Machinery: New York, NY, USA, 2000; pp. 248–251. [\[CrossRef\]](#)
23. Schatzmann, J.; Weilhammer, K.; Stuttle, M.; Young, S. A survey of statistical user simulation techniques for reinforcement learning of dialogue management strategies. *Knowl. Eng. Rev.* **2006**, *21*, 97–126. [\[CrossRef\]](#)
24. Thomaz, A.L.; Breazeal, C. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In Proceedings of the AAAI-06: Twenty-First Conference on Artificial Intelligence, Boston, MA, USA, 16–20 July 2006; pp. 1000–1005.
25. Branavan, S.R.; Chen, H.; Zettlemoyer, L.; Barzilay, R. Reinforcement learning for mapping instructions to actions. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, Singapore, 2–7 August 2009; pp. 82–90.
26. Young, S. Cognitive user interfaces. *IEEE Signal Process. Mag.* **2010**, *27*, 128–140. [\[CrossRef\]](#)
27. Glowacka, D.; Ruotsalo, T.; Konuyshkova, K.; Athukorala, K.; Kaski, S.; Jacucci, G. Directing exploratory search: Reinforcement learning from user interactions with keywords. In Proceedings of the 2013 International Conference on Intelligent User Interfaces, Los Angeles, CA, USA, 19–22 March 2013; pp. 117–128.
28. Littman, M.L. Reinforcement learning improves behaviour from evaluative feedback. *Nature* **2015**, *521*, 445–451. [\[CrossRef\]](#)
29. Debard, Q.; Dibangoye, J.S.; Canu, S.; Wolf, C. Learning 3d navigation protocols on touch interfaces with cooperative multi-agent reinforcement learning. In Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, 16–20 September 2019; Proceedings, Part III. Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 35–52.
30. Li, Z.; Zhao, M.; Das, D.; Zhao, H.; Ma, Y.; Liu, W.; Beaudouin-Lafon, M.; Wang, F.; Ramakrishnan, I.; Bi, X. Select or Suggest? Reinforcement Learning-based Method for High-Accuracy Target Selection on Touchscreens. In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 30 April–5 May 2022; pp. 1–15.
31. Bi, X.; Li, Y.; Zhai, S. Fitts law: Modeling finger touch with fitts' law. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Paris, France, 27 April–2 May 2013; pp. 1363–1372.
32. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. [\[CrossRef\]](#)
33. Gupta, T.; Gori, J. Modeling reciprocal adaptation in HCI: A Multi-Agent Reinforcement Learning Approach. In Proceedings of the Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems, Hamburg, Germany, 23–28 April 2023; pp. 1–6.
34. Li, Z.; Ko, Y.J.; Putkonen, A.; Feiz, S.; Ashok, V.; Ramakrishnan, I.V.; Oulasvirta, A.; Bi, X. Modeling Touch-based Menu Selection Performance of Blind Users via Reinforcement Learning. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, Hamburg, Germany, 23–28 April 2023; pp. 1–18.
35. Sheikh, H.U.; Khadka, S.; Miret, S.; Majumdar, S.; Phielipp, M. Learning intrinsic symbolic rewards in reinforcement learning. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; IEEE: New York, NY, USA, 2022; pp. 1–8. [\[CrossRef\]](#)
36. Amato, C.; Konidaris, G.; Cruz, G.; Maynor, C.A.; How, J.P.; Kaelbling, L.P. Planning for decentralized control of multiple robots under uncertainty. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; IEEE: New York, NY, USA, 2015; pp. 1241–1248. [\[CrossRef\]](#)
37. Ammar, L.B. A usability model for mobile applications generated with a model-driven approach. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 140–146. [\[CrossRef\]](#)
38. Seffah, A.; Donyaee, M.; Kline, R.B.; Padda, H.K. Usability measurement and metrics: A consolidated model. *Softw. Qual. J.* **2006**, *14*, 159–178. [\[CrossRef\]](#)
39. Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE* **2017**, *12*, e0172395. [\[CrossRef\]](#) [\[PubMed\]](#)
40. Verzynov, S.N.; Bochkarev, I.V.; Khramshin, V.R. Development of Line Locator Software Component for Mobile Operating Systems. In *2020 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, 18–22 May 2020*; IEEE: New York, NY, USA, 2020; pp. 1–5. [\[CrossRef\]](#)
41. Chin, J.P.; Diehl, V.A.; Norman, L.K. Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems—CHI '88, Washington, DC, USA, 15–19 May 1988. [\[CrossRef\]](#)
42. Moumane, K.; Idri, A.; Abran, A. Usability evaluation of mobile applications using ISO 9241 and ISO 25062 standards. *SpringerPlus* **2016**, *5*, 548. [\[CrossRef\]](#) [\[PubMed\]](#)
43. Nielsen, J. *Usability Engineering*; Morgan Kaufmann: Burlington, MA, USA, 1994.
44. Liu, W.; Bailly, G.; Howes, A. Effects of frequency distribution on linear menu performance. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, 6–11 May 2017; pp. 1307–1312.
45. Wong, A.; Bäck, T.; Kononova, A.V.; Plaata, A. Deep multiagent reinforcement learning: Challenges and directions. *Artif. Intell. Rev.* **2023**, *56*, 5023–5056. [\[CrossRef\]](#)

46. Oroojlooy, A.; Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *Appl. Intell.* **2023**, *53*, 13677–13722. [[CrossRef](#)]
47. Carta, T.; Romac, C.; Wolf, T.; Lamprier, S.; Sigaud, O.; Oudeyer, P.-Y. Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning. In Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; pp. 3676–3713.
48. Li, W.; Qiao, D.; Wang, B.; Wang, X.; Jin, B.; Zha, H. Semantically Aligned Task Decomposition in Multi-Agent Reinforcement Learning. *arXiv* **2023**, arXiv:2305.10865. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.