Generating Adaptive Route Instructions Using Hierarchical Reinforcement Learning

Heriberto Cuayáhuitl[1], Nina Dethlefs[2], Lutz Frommberger[1], Kai-Florian Richter[1], and John Bateman[1,2]

[1] Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition, University of Bremen
Enrique-Schmidt-Str. 5, 28359, Bremen, Germany
[2] FB10 Faculty of Linguistics and Literary Sciences, University of Bremen
Bibliothekstrasse 1, 28359, Bremen, Germany
heriberto@uni-bremen.de

Abstract. We present a learning approach for efficiently inducing adaptive behaviour of route instructions. For such a purpose we propose a two-stage approach to learn a hierarchy of wayfinding strategies using hierarchical reinforcement learning. Whilst the first stage learns low-level behaviour, the second stage focuses on learning high-level behaviour. In our proposed approach, only the latter is to be applied at runtime in user-machine interactions. Our experiments are based on an indoor navigation scenario for a building that is complex to navigate. We compared our approach with flat reinforcement learning and a fully-learnt hierarchical approach. Our experimental results show that our proposed approach learns significantly faster than the baseline approaches. In addition, the learnt behaviour shows to adapt to the type of user and structure of the spatial environment. This approach is attractive to automatic route giving since it combines fast learning with adaptive behaviour.

1 Introduction

Adaptive route instructions provide a human user with information that is specifically suitable to his or her needs by taking the current situation and individual properties of the user into account. Such kind of adaptation is achieved along two dimensions: (a) the cognitive principles underlying human route descriptions and (b) models of different user types who may differ in their informational needs. The first aspect is related to what Lovelace et al. [1] stated as characteristics of good route descriptions, which are based on experimental findings involving human route descriptions. These include the preference of landmarks over metrical distances or purely path-based information, the inclusion of confirmatory information on long route segments to prevent the user from going too far, and limiting the amount of detail to a necessary minimum to avoid redundancy. The second aspect is related to the fact that users may differ in their amount of prior knowledge and hence in their informational needs or preferences. For example,

C. Hölscher et al. (Eds.): Spatial Cognition VII, LNAI 6222, pp. 319–334, 2010.

a user who is (partially) familiar with an environment might prefer the shortest route to some destination, while an unfamiliar user might prefer the easiest route in order not to get lost. Similarly, while the first type of user might be able to find their way with a limited amount of detail and perceive too much redundancy as disturbing or patronizing, the latter type of user will likely require more guidance.

In this paper, we address the topic of generating online adaptive route instructions using *reinforcement learning* (RL) [2]. In order to achieve adaptive route instructions while keeping the system efficient, as it is needed in navigation systems, for example, we propose to apply a two-stage approach to learn a hierarchy of wayfinding strategies employing *hierarchical reinforcement learning* (HRL). In a first stage, we learn low-level behaviour, such as 'left' or 'right.' In a second stage, we learn high-level behaviour, such as 'go to the next junction.' The latter behaviour can be applied in online user-machine interactions to provide online adaptation and interactive route guidance. This enables the system to identify a user's prior knowledge and information needs during an interaction and to flexibly adapt to them. We present experimental results, based on a simulated environment, that compare flat RL with HRL fully-online and semi-online behaviour. We demonstrate that the semi-online policies are learnt faster and more efficiently than the others and hence are suitable for online learning and adaptation.

This paper is organized as follows: Section 2 gives an overview of related work on adaptive route generation. Sections 3 and 4 introduce the approaches of reinforcement learning and hierarchical reinforcement learning, respectively, while the performance of fully-online and semi-online learnt behaviours are compared in Section 5. In that section, we will show that our proposed approach is promising for its application in online user-machine interaction and that it can be used for inducing adaptive behaviour in in-situ navigation assistance.

2 Related Work on Route Instruction Generation

Related work on generation of adapative route instructions has addressed several issues, including the usage of landmarks, cognitive adaquacy of route descriptions in changing spatial situations, and the tailoring of route descriptions towards different user groups. Landmarks are highly prominent features in route directions [3], both because they are crucial elements in structuring human knowledge about an environment [4] and because they are powerful means to link wayfinding actions to the points along a route where they have to be performed, i.e., in providing procedural information [1]. In general human direction givers try to provide concise instructions, in order not to overload the receiver with too much information [5]. The procedural nature of landmarks makes them essential elements in human route directions, which can also be attributed to cognitive landmark-based mechanisms of combining several consecutive instructions into a single, higher-order instruction, which is more compact (so called spatial chunking, [6]). Not surprisingly, landmarks were identified as one key feature of improving automated navigation services [7, 8]. However, today's commercially

available navigation services hardly include any landmarks at all. They produce instructions in a uniform, repetitive form, referring to street names and metric distances only.

In addition to the inclusion of landmarks in route directions, a further issue of major relevance in the context of this paper is adaptation to changing spatial situations. As pointed out by Klippel et al. [9], people vary descriptions on what to do at intersections to both what action has to be performed (a functional aspect) and to the layout of the intersection, including the presence of landmarks (a structural aspect). More generally, people adapt instructions to the assumed recipients' information needs, which includes taking into account the transportation means and the required detail (scale) to locate the destination (e.g., an ATM mounted on a wall of a bank compared to the bank building itself) [10]. Attempts to capture this need for adaptation in automated services were pursued on different levels. Some approaches aimed for adapting the underlying route to travel between origin and destination by searching for the simplest path [11], the most reliable path, i.e., the one with the least probability for wayfinding errors [12], or the path that is the simplest to describe [13].

Systems that have attempted to integrate several of the abovementioned principles include Dale et al.'s CORAL system [14]. It uses data that is available from existing geographical databases to identify relevant information for describing a route, which includes features that may serve as landmarks. The authors apply natural language generation methods to construct instructions that adapt to the given environment. Namely, these methods are segmentation (comparable to spatial chunking), which groups single instructions into higher-level instructions, and aggregation, which is a process to construct sentences that communicate several pieces of information at once. A similar approach was presented by Richter [15]. His work focused on the steps before the actual information presentation, i.e., the identification of relevant information and spatial chunking. It includes methods for determining the functional roles of different kinds of landmark candidates, among them point-like, linear, area-like features, as well as intersection structure (cf. [16]), and an optimization process to find the 'best' combination of chunks (e.g., the most concise description of a route).

Research that applied machine learning to route generation includes work by Marciniak and Strube [17, 18], who used machine learning techniques to generate natural language route directions. Using a manually annotated corpus of existing route directions as input, they had their system learn K* classifiers [19] for individual elements of a vector representation of an expression's form. Overall expression generation is then realized as a cascade of these classifiers, where later classifiers can make use of the output of earlier classifiers. Expressions are represented as tree adjoining grammars (TAGs).

We are not aware of any prior work that has applied reinforcement learning to adaptive route instruction generation. In the rest of the paper we show that hierarchical reinforcement learning (based on the divide-and-conquer approach) is a promising method to optimize—in an efficient and scalable way—the behaviour of adaptive route instruction generation systems.

3 Reinforcement Learning for Route Instruction Generation

3.1 Adapting to a User's Behaviour with Reinforcement Learning

Reinforcement learning is particularily useful if the characteristics of an underlying system are unknown or only partially known. This is what we encounter when generating adaptive route instructions. Generated route instructions so far make assumptions on how a user can cope with the produced directions. While these assumptions may hold for a majority of users, it does not take special skills or preferences of an individuum into account. For example, a user familiar with an environment will prefer different instructions than someone visiting this place for the first time. Presented with instructions aiming at the experts' knowledge, the unexperienced user will most likely fail to follow the instructions correctly. Other users may just have shifted preferences (like counting crossroads or looking for landmarks). These preferences can be assumed, but usually not fully described, and are a matter of observation of the user's interaction with the environment.

This observed interaction is the fundament of reinforcement learning, as the learned behaviour (the *policy*) is adapted by rewards depending on actions agents perform within their environment. RL is able to produce an optimal policy for a system that cannot be described in a closed form simply by interacting with it. Thus, it becomes a valuable means for generating adaptive route instructions.

3.2 Formalization of the Learning Task

A user-machine interaction for route instruction generation can be described as follows: The machine receives a user input (e.g., a query requesting instructions or the user's current location) to navigate from an origin to a destination. It enters such information into its knowledge base and then extracts an environment state s to choose an instruction (also referred to as 'action' a). Then, the chosen action is used to generate the corresponding output conveyed to the user. These kinds of interactions are followed in an iterative process until the goal is reached. Assuming that the machine receives a numerical reward r for executing action a when the spatial environment makes a transition from state s_t (at time t) to a next state s_{t+1}, an interaction can be expressed as $\{s_1, a_1, r_2, s_2, a_2, r_3, ..., s_{T-1}, a_{T-1}, r_T, s_T\}$, where T is the final time step.

Such sequences can be used by a reinforcement learning agent to optimize the behaviour of the route instruction controller. A reinforcement learning wayfinding agent aims to learn its behaviour from interaction with an environment, where situations are mapped to actions by maximizing a long-term reward signal. The standard reinforcement learning paradigm makes use of the formalism of Markov Decision Processes (MDPs) [2]. An MDP is characterized by a set of states S, a set of actions A, a state transition function T, and a reward or performance function R that rewards the agent for each selected action. A given sequence of states, actions, and rewards receives a total cumulative discounted reward expressed as

$$r = r_1 + \gamma r_2 + \gamma^2 r_3 + ...\gamma^{\tau-1} r_\tau = \sum_{k=0}^{\tau-1} \gamma^k r_{k+1}, \tag{1}$$

where the discount rate $0 \leq \gamma \leq 1$ makes future rewards less valuable than immediate rewards as it approaches 0. In the equation above, the term on the right-hand side is referred to as 'the expected value of the reward,' and can be computed recursively using a state-value function $V^\pi(s)$, which returns the value of starting in state s and then following policy π thereafter. The value-function is defined by the Bellman equation for V^π expressed as

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P(s'|s,a) \left[R(s'|s,a) + \gamma V^\pi(s')\right]. \tag{2}$$

Alternatively, the expected value of the reward can also be computed recursively using an action-value function $Q^\pi(s,a)$, which returns the cumulative reward of starting in state s, taking action a and then following policy π thereafter. The action-value function is defined by the Bellman equation for Q^π expressed as

$$Q^\pi(s,a) = \sum_{s'} P(s'|s,a) \left[R(s'|s,a) + \gamma V^\pi(s')\right]. \tag{3}$$

An optimal policy π^* can be found by using the following Bellman equations that represent a system of equations, one for each state:

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a) \left[R(s'|s,a) + \gamma V^*(s')\right], \tag{4}$$

or state-action pair:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a) \left[R(s'|s,a) + \gamma \max_{a'} Q^*(s',a')\right]. \tag{5}$$

Finally, an optimal policy performs action selection according to

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s,a). \tag{6}$$

The optimal policy π^* can be found using reinforcement learning algorithms such as Q-learning or SARSA [2]. However, the flat tabular reinforcement learning framework lacks scalability, that is, it requires very long training times as the problems become more complex and learning becomes infeasible for anything but very small state spaces [20]. But for generating online route instructions we must rely on fast training times and immediate adaptations. Thus, we use a hierarchical reinforcement learning framework that decomposes an MDP into a hierarchy of Semi-Markov decision processes (each one representing a subsequence of instructions). This approach has been successfully applied to spoken dialogue agents with large state spaces [21, 22].

4 Hierarchical Reinforcement Learning Approaches for Route Instruction Generation

In this research we treat the task of route instruction generation as a discrete Semi-Markov Decision Process (SMDP) in order to address the problem of scalable optimization. A discrete-time SMDP $M = <S, A, T, R>$ is characterized by a set of states S; a set of actions A; a transition function T that specifies the next state s' given the current state s and action a with probability $P(s', \tau|s, a)$; and a reward function $R(s', \tau|s, a)$ that specifies the reward given to the agent for choosing action a when the environment makes a transition from state s to state s'. The random variable τ denotes the number of time-steps taken to execute action a in state s. This formulation allows temporal abstraction, where actions take a variable amount of time to complete their execution [23]. In this model two types of actions can be distinguished: (a) primitive actions roughly corresponding to low-level instructions (e.g., straight, right, left, turn around), and (b) composite actions corresponding to sub-sequences of instructions (e.g., go to the other end of the corridor). Whilst the root and children models execute primitive and composite actions, the grandchildren models—at the bottom of the hierarchy—execute only primitive actions.

4.1 Fully-Online Hierarchical Learning

This research treats each composite action as a separate SMDP as described in Cuayáhuitl [21]. In this way an MDP can be decomposed into multiple SMDPs hierarchically organized into L levels and N models per level. We denote each SMDP as $\mathcal{M} = \{M_j^i\}$, where $j \in \{0, ..., N-1\}$ and $i \in \{0, ..., L-1\}$. Thus, any given SMDP in the hierarchy is denoted as $M_j^i = <S_j^i, A_j^i, T_j^i, R_j^i>$, see Figure 1 for an illustration. The indexes i and j only identify an SMDP in a unique way in the hierarchy, they do not specify the execution sequence of SMDPs because that is learnt by the reinforcement learning agent.

The goal in an SMDP is to find an optimal policy π^* that maximizes the reward of each visited state. The optimal action-value function $Q^*(s, a)$ specifies the expected cumulative reward for executing action a in s and then following π^*. The Bellman equations for Q^* of subtask M_j^i can be expressed as

$$Q_j^{*i}(s, a) = \sum_{s', \tau} P_j^i(s', \tau|s, a)[R_j^i(s', \tau|s, a) + \gamma^\tau \max_{a'} Q_j^{*i}(s', a')], \tag{7}$$

The optimal policy for each model in the hierarchy is defined by

$$\pi_j^{*i}(s) = \arg\max_{a \in A_j^i} Q_j^{*i}(s, a). \tag{8}$$

These policies can be found by reinforcement learning algorithms for SMDPs such as the Hierarchical Semi-Markov Q-Learning algorithm (also referred to as HSMQ-Learning) [24]. In this algorithm, the action-value function Q_j^{*i} of equation 7 is approximated according to

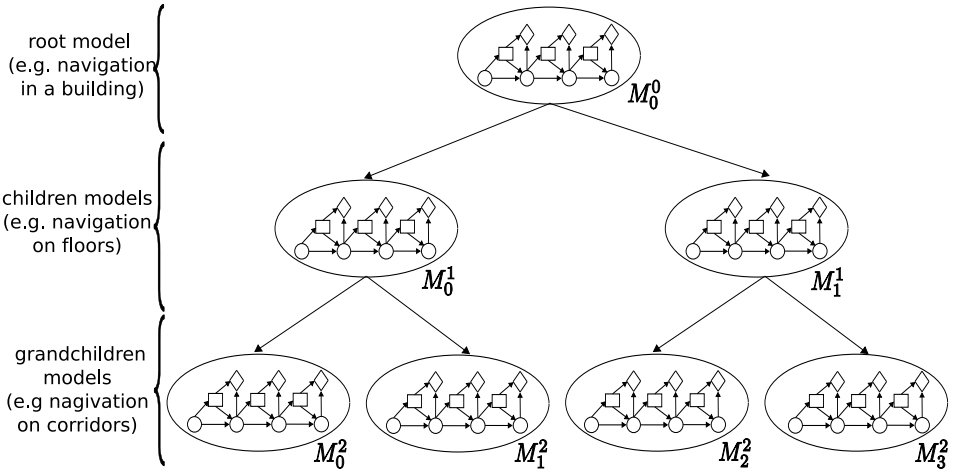

Fig. 1. Hierarchy of SMDPs M_j^i, where i denotes a level and j the model per level. In each SMDP, bottom circles represent environment states, rectangles represent actions, and diamonds represent rewards.

$$Q_j^i(s_t, a_t) \leftarrow (1-\alpha)Q_j^i(s_t, a_t) + \alpha\left[r_{t+\tau} + \gamma^\tau \max_{a'} Q_j^i(s_{t+\tau}, a')\right]. \quad (9)$$

The summation over all τ time steps as they appear in equation 7 is reflected here by using cumulative rewards $r_{t+\tau} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{\tau-1} r_{t+\tau}$ received for executing actions a_t, and by raising γ to the power τ.

4.2 Semi-online Hierarchical Learning

Because efficient learning is required at runtime (i.e., users should not wait too long for receiving route instructions), we propose a two-stage approach to speed up the hierarchical method above. The first stage—applied before user-machine interaction—induces the behaviour of the policies at the bottom of the hierarchy, expressed as

$$\pi_j^{*L-1}(s) = \arg\max_{a \in A_j^{L-1}} Q_j^{*L-1}(s, a), \quad (10)$$

where the index $L-1$ represents the bottom layer in the hierarchy. The second stage—applied at runtime user-machine interaction—induces the behaviour of parent agents and the bottom agent including the goal state, expressed as

$$\pi_j^{*i}(s) = \arg\max_{a \in A_j^i} Q_j^{*i}(s, a), \forall i \neq L-1 \vee s^g \in S_j^i, \quad (11)$$

where s^g is the goal state. Our approach avoids learning from scratch by reusing learnt behaviour from the first stage, and focuses on learning high-level behaviour. The same learning algorithm can also be used to find the policies above.

5 Experiments and Results

The aim of our experiments was: (1) to provide a proof-of-concept application of our semi-online learning approach by comparing it against fully-online flat and hierarchical learning, and (2) to induce adaptive behaviour for familiar and unfamiliar users. Our experiments are based on indoor route instruction generation, and use a simulated spatial environment with data derived from a real building that is complex to navigate. We employ data from a single floor, which is represented as an indirected graph (see Fig. 2). Such a graph and the stochastic behaviour described in subsection 5.2 form the agent's learning environment.

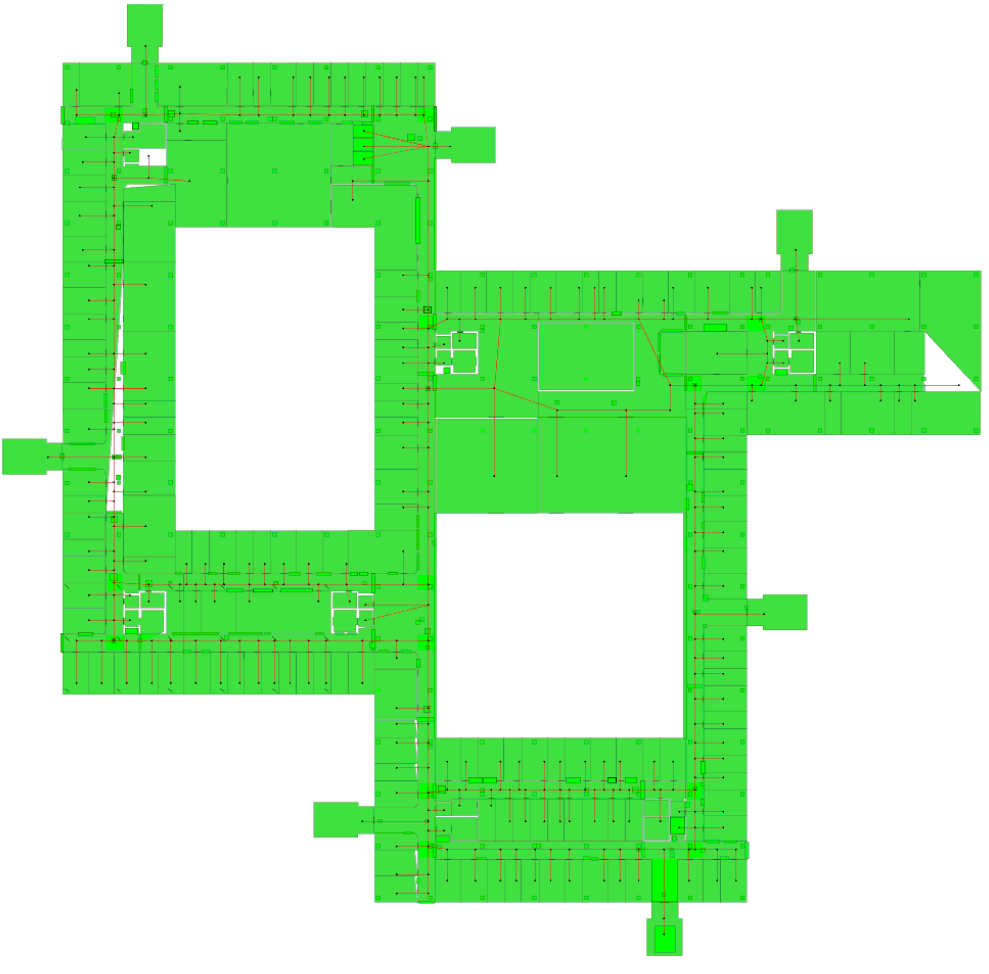

Fig. 2. Map showing an undirected graph of the spatial data used for learning navigational behaviour. The small black circles represent graph nodes, which are connected with the corresponding edges so that navigation is allowed between corridors and rooms.

5.1 Experimental Setting

Flat reinforcement learning used the state space representation shown in Table 1, and the action set shown in Table 2, resulting in a space of 896000 state-actions. Whilst the first two state variables (location and orientation) represent basic information that can be used for local navigational decisions, the other state variables (score of best landmark, segment length, segments to follow, and user type) represent information of the route trajectory that can be used for more informative navigational decisions. The latter can take into account the inclusion or exclusion of landmarks and using a trajectory that is less confusing to follow according to the given stochastic environment.

The reward function addresses compact instructions based on the shortest number of turns. It is defined by the following rewards given to the reinforcement learning agent for choosing action a when the environment makes a transition from state s to state s':

$$r(s,a,s') = \begin{cases} 0 \text{ for reaching the goal state} \\ \text{-1 for a straight action.} \\ \text{-10} \quad \text{otherwise.} \end{cases} \tag{12}$$

For hierarchical learning, the state-action space representation used 82 models (one parent and 81 children) induced automatically from turning points in the spatial environment. Whilst the root agent used child agents instead of the action

Table 1. State variables for generating indoor route instructions. Flat learning used a graph with 400 nodes, corresponding to $|S| = 400 * 4 * 5 * 2 * 4 * 2 = 128000$ states. We follow Raubal & Winter [25] to rank landmarks based on the categories suggested by Sorrows & Hirtle [4].

Variable	Values	Description
Location	(x,y)	Coordinates of graph nodes from the space
Orientation	$\{0,1,2,3\}$	Corresponding to 'south', 'north', 'east', 'west'
Score of best landmark	$\{0,1,2,3,4\}$	The higher the more suitable
Segment length	$\{0,1\}$	Corresponding to 'short' and 'long'
Segments to follow	$\{0,1,2,3\}$	Number of segments in the location
User type	$\{0,1\}$	Corresponding to 'unfamiliar' and 'familiar'

Table 2. Primitive actions for generating indoor route instructions

Action	Description
straight	go straight
left_nolandmark	turn left without referring to a present landmark
left_landmark	turn left referring to a present landmark
right_nolandmark	turn right without referring to a present landmark
right_landmark	turn right referring to a present landmark
turnaround_nolandmark	turn around without referring to a present landmark
turnaround_landmark	turn around referring to a present landmark

'straight', the child agents used the same action set as the flat learning agent. In addition, these experiments used the same state transition and reward functions as in flat learning, used in each agent.

The learning setup used Q-Learning for flat reinforcement learning [2] and HSMQ-Learning for hierarchical reinforcement learning (briefly described in the previous section). The learning parameters used by the algorithms were the same for all learning approaches. The learning rate parameter α decays from 1 to 0 according to $\alpha = \frac{100}{(100+\tau)}$, where τ represents elapsed time-steps in the current subtask. Each subtask M_j^i had its own learning rate. The discount factor $\gamma = 1$ makes future rewards as valuable as immediate rewards. The action selection strategy used ϵ-Greedy with $\epsilon = 0.01$, and initial Q-values of 0.

5.2 Stochastic Behaviour in the Learning Environment

The conditional probabilities shown in Table 3 were used to model user confusion in indoor wayfinding. Such probabilities were hand-crafted according to the following assumptions.

- Unfamiliar users get more confused than familiar users.
- Goal destinations within short segments (corridors) are easier to find than within long segments.
- Users get more confused at junctions with three or more segments than with two or less segments.
- The confusion is lower when following suitable landmarks (score ≥ 3), and higher when following less suitable landmarks (score ≤ 2).

Finally, because the probabilities above are location-independent for the spatial environment, we used additional location-dependent probabilities expressed as

$$Pr(UserConfusion|TurningPoint, Orientation, UserType) \qquad (13)$$

for modelling user confusion, specifically at turning points.

5.3 Experimental Results

Figure 3 shows the learning curves of policies for an average-sized route instruction, averaged over 10 training runs of 10^4 episodes. It can be observed that our proposed approach learned faster than the other approaches. Whilst the latter two require four orders of magnitude (10^4 episodes), our proposed approach required only three orders of magnitude (10^3 episodes). The fact that our proposed approach stabilizes its behaviour much more quickly than the others suggests that it has promising application for online user-machine interaction. To give an idea of learning times, the instructions shown in Table 4 took 3.2 seconds to generate using 1000 episodes on a MacBook computer (processor Intel Core 2 Duo of 2.4 GHz and 2GB in RAM).

The first column of the sample set shown in Table 4 shows the navigation policies π_j^{*i} used by the hierarchical learning agent, where index j of the lower

Table 3. Conditional probability table to model user confusion in wayfinding. The first four columns are state variables (see Fig. 1 for a description of their values) and the last two columns are state transition probabilities. Notation: s = current environment state, $\tilde{s'}$ =distorted next state, a_{wl} =action with landmark, a_{wol} =action without landmark.

User Type	Segment Length	Segments to Follow	Landmark Weight	$P(\tilde{s'}\|s, a_{wl})$	$P(\tilde{s'}\|s, a_{wol})$
0	0	≤ 2	≤ 2	.10	.15
0	0	≤ 2	≥ 3	.10	.10
0	0	≥ 3	≤ 2	.10	.25
0	0	≥ 3	≥ 3	.10	.20
0	1	≤ 2	≤ 2	.10	.15
0	1	≤ 2	≥ 3	.10	.10
0	1	≥ 3	≤ 2	.10	.35
0	1	≥ 3	≥ 3	.10	.30
1	0	≤ 2	≤ 2	.05	.10
1	0	≤ 2	≥ 3	.05	.05
1	0	≥ 3	≤ 2	.05	.15
1	0	≥ 3	≥ 3	.05	.10
1	1	≤ 2	≤ 2	.05	.10
1	1	≤ 2	≥ 3	.05	.05
1	1	≥ 3	≤ 2	.05	.20
1	1	≥ 3	≥ 3	.05	.15

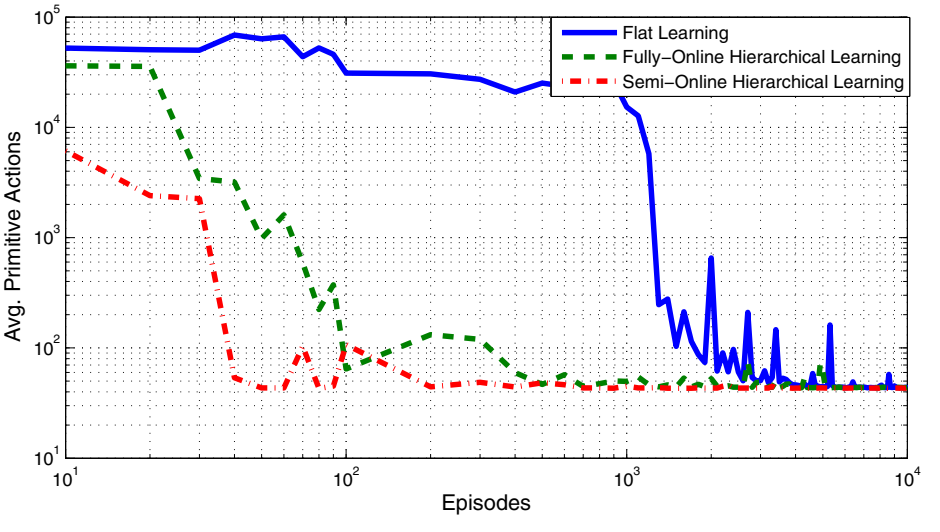

Fig. 3. Learning curves of route instruction strategies with three learning approaches. We plot the average number of primitive actions because our learning agents aim for efficient trajectories according to the given stochastic environment. Alternatively, other measures can be used, such as average number of textual route instructions.

Table 4. Sample generation of route instructions using a hierarchy of policies π_j^{*i}, where π_0^{*0} is the root policy and π_j^{*1} are the children policies. Whilst the former is used to navigate from origin to destination, the latter are used to navigate across segments. The term 'segment' used here denotes a route with a start and end junctions, but without intermediate junctions.

Policy	Environment State	Actions	Interpretation
π_0^{*0}	(2250,1330),3,0,0,0,0	$\pi_{307-314}^{*1}$	Go straight until
$\pi_{307-314}^{*1}$	(2250,1330),3,0,0,0,0	straight (3 times)	the second junction.
π_0^{*0}	(1690,1260),3,1,0,3,0	$\pi_{314-192}^{*1}$	
$\pi_{314-192}^{*1}$	(1690,1260),3,1,0,3,0	straight	
π_0^{*0}	(2250,1330),3,0,0,0,0	left_landmark	Turn left at <landmark>
π_0^{*0}	(1480,1260),0,1,1,3,0	$\pi_{192-1000}^{*1}$	and go straight until
$\pi_{192-1000}^{*1}$	(1480,1260),0,1,1,3,0	straight (six times)	the next junction.
π_0^{*0}	(1480,1885),0,2,1,3,0	right_landmark, $\pi_{1000-110}^{*1}$	Turn right at <landmark>
$\pi_{1000-110}^{*1}$	(1480,1885),3,3,1,3,0	straight (15 times)	and go straight until
			the next junction.
π_0^{*0}	(480,1880),3,1,1,3,0	right_landmark, π_{110-89}^{*1}	Turn right at <landmark>
π_{110-89}^{*1}	(480,1880),1,3,1,3,0	straight (9 times)	and go straight until
π_0^{*0}	(480,1480),1,1,1,3,0	π_{89-15}^{*0}	<landmark>, the destination
π_{89-15}^{*0}	(480,1480),1,1,1,3,0	straight (4 times)	will be at your right.
		right_landmark, straight	
π_0^{*0}	(580,1260),2,1,1,0,0	[goal state]	

level of the hierarchy represents a route segment with graph nodes 'start-end'. For example, policy $\pi_{192-1000}^{*1}$ navigates from node 192 to node 1000. The second column shows the environment state using the state variables described in Table 1. The third column shows the actions taken by the agent; recall that whilst the hierarchical learning agents take primitive actions a and composite actions π_j^{*i}, the flat learning agent only takes primitive actions. The fourth and final column provides an interpretation of chunked route instructions, where the chunking is based on major changes of direction such as left or right.

Our preliminary results show that the learnt policies include more landmarks for unfamiliar users, which would result in longer natural language instructions that provide a better anchoring of actions in the environment. In addition, we observed that the learnt behaviour can provide tailored instructions for user types. For example, the partial map shown in Figure 4 shows two routes, one for each type of user. Whilst the learnt route for the familiar user followed the trajectory 307, 306, 315, 437, 356, 363, 364, the learnt route for the unfamiliar user followed the trajectory 307, 314, 192, 1000, 239, 227, 363, 364. The former route included 28 nodes and is shorter than the latter with 34 nodes, but the latter was found easier to navigate, avoiding user confusions in the stochastic environment.

Finally, additional stochastic behaviour can be incorporated in the spatial environment in order to support other forms of adaptive behaviour. For example, the learning agent can ask the user for about intermediate locations, where

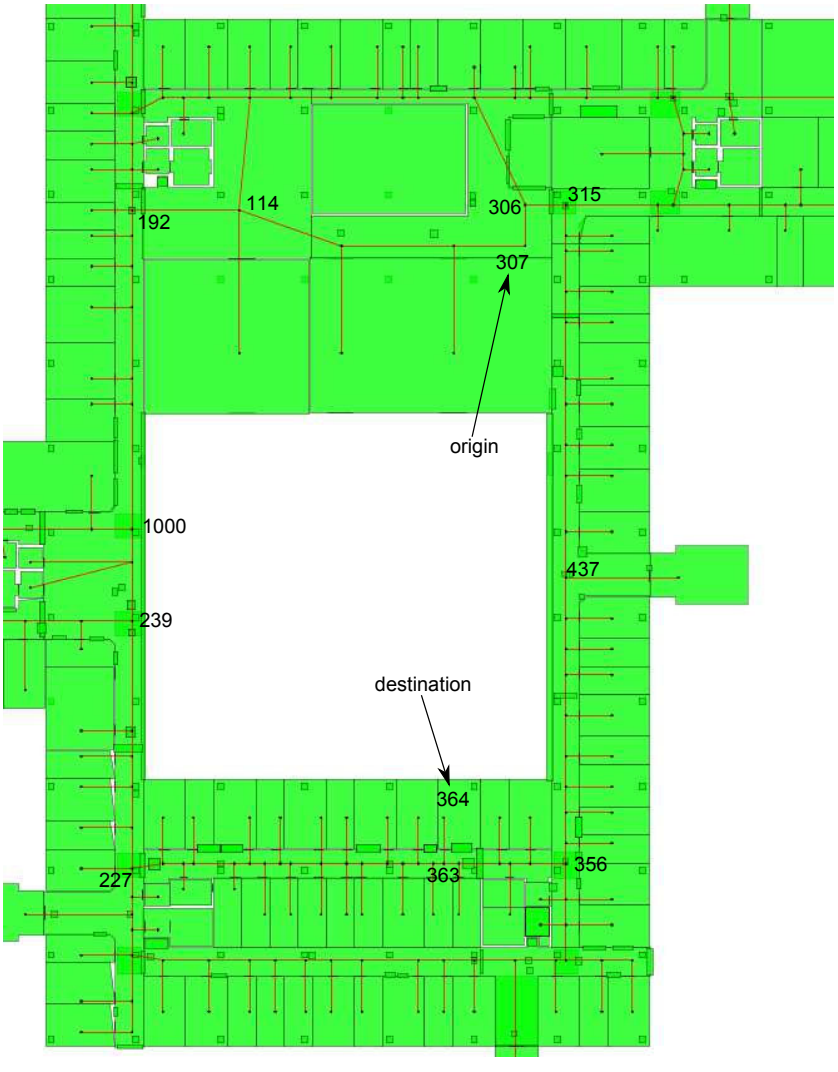

Fig. 4. Partial map illustrating an example of adaptation to the user type: familiar and unfamiliar. Whilst the learnt route for the familiar user followed the trajectory 307, 306, 315, 437, 356, 363, 364, the learnt route for the unfamiliar user followed the trajectory 307, 314, 192, 1000, 239, 227, 363, 364.

each location can incorporate a probability distribution of being known. This is reasonable if we assume that users have partial knowledge of the spatial environment. Intermediate locations may then serve as alternative starting points for providing route directions (e.g., "if you know the main staircase, go there first"). Such probabilistic knowledge can be derived from prior knowledge or learnt from data. The proposed reinforcement learning approach can be used to derive a unified set of optimized behaviours.

6 Conclusions and Future Work

We have presented an approach for fast reinforcement learning of route instruction strategies. This approach decomposes a problem into subproblems in order to learn low-level behaviour in advance (i.e., before user-machine interaction), and at run-time it learns high-level behaviour. We evaluated our approach in a simulated environment for indoor wayfinding, and compared it against flat reinforcement learning and hierarchical fully-learnt behaviour. We found that whilst the latter two require four orders of magnitude (10^4 episodes) to learn the behaviour of an average-sized route, our proposed approach required only three orders of magnitude (10^3 episodes). This result suggests that our method is suitable for its application in online user-machine interactions. Furthermore, the learnt policies induced different behaviour according to different given probabilistic state transition function that modelled user confusions in wayfinding. Thus, the proposed system is able to produce user-adaptive route instructions.

Suggested future work is as follows:

- To evaluate our proposed approach in an environment with real users. This is necessary in order to assess the agents performance in a quantitative and qualitative way. For such a purpose, our proposed approach can be integrated into a (spoken) dialogue system with natural language input and output, as described in [26].
- To investigate how to guide (confused) users in a dialogic interaction, where adaptive guidance would be essential for successful wayfinding. Such behaviour could address the issue of learning to negotiate route instructions. This makes reinforcement learning attractive to wayfinding systems because it optimizes decision-making given the history of the interaction represented in the environment state.
- To test different state representations and reward functions, for instance, the performance function induced from wayfinding dialogues described in [27].
- In a next step we plan to employ techniques of spatial abstraction to include structural knowledge into the state representation [28]. This would allow for another speed-up in learning and for reusing gained knowledge in previously unexplored situations [29]. This is especially useful if the optimization of a hierarchical agent is intractable (i.e., the state-action space becomes too large and is indecomposable).

Acknowledgements

Part of this work was supported by the Transregional Collaborative Research Center SFB/TR 8 "Spatial Cognition." Funding by the German Research Foundation (DFG) is gratefully acknowledged.

References

[1] Lovelace, K.L., Hegarty, M., Montello, D.R.: Elements of good route directions in familiar and unfamiliar environments. In: Freksa, C., Mark, D.M. (eds.) COSIT 1999. LNCS, vol. 1661, pp. 65–82. Springer, Heidelberg (1999)
[2] Sutton, R., Barto, A.: Reinforcement Learing: An Introduction. MIT Press, Cambridge (1998)
[3] Denis, M.: The description of routes: A cognitive approach to the production of spatial discourse. Cahiers Psychologie Cognitive 16(4), 409–458 (1997)
[4] Sorrows, M.E., Hirtle, S.C.: The nature of landmarks for real and electronic spaces. In: Freksa, C., Mark, D.M. (eds.) COSIT 1999. LNCS, vol. 1661, pp. 37–50. Springer, Heidelberg (1999)
[5] Daniel, M.P., Denis, M.: The production of route directions: investigating conditions that favour conciseness in spatial discourse. Applied Cognitive Psychology 18(1), 57–75 (2004)
[6] Klippel, A., Hansen, S., Richter, K.F., Winter, S.: Urban granularities - a data structure for cognitively ergonomic route directions. GeoInformatica 13(2), 223–247 (2009)
[7] May, A.J., Ross, T., Bayer, S.H., Burnett, G.: Using landmarks to enhance navigation systems: Driver requirements and industrial constraints. In: Proceedings of the 8th World Congress on Intelligent Transport Systems, Sydney, Australia (2001)
[8] Ross, T., May, A., Thompson, S.: The use of landmarks in pedestrian navigation instructions and the effects of context. In: Brewster, S., Dunlop, M. (eds.) Mobile HCI 2004. LNCS, vol. 3160, pp. 300–304. Springer, Heidelberg (2004)
[9] Klippel, A., Tenbrink, T., Montello, D.R.: The role of structure and function in the conceptualization of directions. In: van der Zee, E., Vulchanova, M. (eds.) Motion Encoding in Language and Space. Oxford University Press, Oxford (to appear)
[10] Tenbrink, T., Winter, S.: Variable granularity in route directions. Spatial Cognition and Computation: An Interdisciplinary Journal 9(1), 64–93 (2009)
[11] Duckham, M., Kulik, L.: "Simplest" paths: Automated route selection for navigation. In: Kuhn, W., Worboys, M., Timpf, S. (eds.) COSIT 2003. LNCS, vol. 2825, pp. 169–185. Springer, Heidelberg (2003)
[12] Haque, S., Kulik, L., Klippel, A.: Algorithms for reliable navigation and wayfinding. In: Barkowsky, T., Knauff, M., Ligozat, G., Montello, D.R. (eds.) Spatial Cognition 2007. LNCS (LNAI), vol. 4387, pp. 308–326. Springer, Heidelberg (2007)
[13] Richter, K.F., Duckham, M.: Simplest instructions: Finding easy-to-describe routes for navigation. In: Cova, T.J., Miller, H.J., Beard, K., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2008. LNCS, vol. 5266, pp. 274–289. Springer, Heidelberg (2008)
[14] Dale, R., Geldof, S., Prost, J.P.: Using natural language generation in automatic route description. Journal of Research and Practice in Information Technology 37(1), 89–105 (2005)

[15] Richter, K.F.: Context-Specific Route Directions - Generation of Cognitively Motivated Wayfinding Instructions. DisKi 314 / SFB/TR 8 Monographs, vol. 3. IOS Press, Amsterdam (2008)
[16] Klippel, A., Richter, K.F., Hansen, S.: Structural salience as a landmark. In: Workshop Mobile Maps 2005, Salzburg, Austria (2005)
[17] Marciniak, T., Strube, M.: Classification-based generation using TAG. In: Belz, A., Evans, R., Piwek, P. (eds.) INLG 2004. LNCS (LNAI), vol. 3123, pp. 100–109. Springer, Heidelberg (2004)
[18] Marciniak, T., Strube, M.: Modeling and annotating the semantics of route directions. In: Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-6), Tilburg, The Netherlands, pp. 151–162 (2005)
[19] Cleary, J., Trigg, L.: An instance-based learner using an entropic distance measure. In: Proceedings of the 12th International Conference on Machine Learning, Tahoe City, Ca, pp. 108–114 (1995)
[20] Kaelbling, L.P., Littmann, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research 4, 237–285 (1996)
[21] Cuayáhuitl, H.: Hierarchical Reinforcement Learning for Spoken Dialogue Systems. PhD thesis, School of Informatics, University of Edinburgh (January 2009)
[22] Cuayáhuitl, H., Renals, S., Lemon, O., Shimodaira, H.: Evaluation of a hierarchical reinforcement learning spoken dialogue system. Computer Speech and Language 24(2), 395–429 (2010)
[23] Dietterich, T.: Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research 13(1), 227–303 (2000)
[24] Dietterich, T.: An overview of MAXQ hierarchical reinforcement learning. In: Choueiry, B.Y., Walsh, T. (eds.) SARA 2000. LNCS (LNAI), vol. 1864, pp. 26–44. Springer, Heidelberg (2000)
[25] Raubal, M., Winter, S.: Enriching wayfinding instructions with local landmarks. In: Egenhofer, M., Mark, D. (eds.) GIScience 2002. LNCS, vol. 2478, pp. 243–259. Springer, Heidelberg (2002)
[26] Cuayáhuitl, H., Dethlefs, N., Richter, K.F., Tenbrink, T., Bateman, J.: A dialogue system for indoor wayfinding using text-based natural language. In: Proceedings of the 11th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), Iasi, Romania (2010)
[27] Dethlefs, N., Cuayáhuitl, H., Richter, K.F., Andonova, E., Bateman, J.: Evaluating task success in a dialogue system for indoor navigation. In: Proc. of the 14th Workshop on the Semantics and Pragmatics of Dialogue, SemDial (2010)
[28] Frommberger, L., Wolter, D.: Spatial abstraction: Aspectualization, coarsening, and conceptual classification. In: Freksa, C., Newcombe, N.S., Gärdenfors, P., Wölfl, S. (eds.) Spatial Cognition VI. LNCS (LNAI), vol. 5248, pp. 311–327. Springer, Heidelberg (2008)
[29] Frommberger, L.: Situation dependent spatial abstraction in reinforcement learning. In: ICML/UAI/COLT Workshop on Abstraction in Reinforcement Learning, Montreal, Canada (2009)