HPLAN: Facilitating the Implementation of Joint Human-Agent Activities

Sebastian Ahrndt, Philipp Ebert, Johannes Fähndrich, and Sahin Albayrak

DAI-Laboratory, Technische Universität Berlin,
Faculty of Electrical Engineering and Computer Science,
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
sebastian.ahrndt@dai-labor.de (Corresponding author)

Abstract. When it comes to planning for joint human-agent activities, one has to consider not only flexible plan execution and social constraints but also the dynamic nature of humans. This can be achieved by providing additional information about the characteristics of a human. As an example one need to take the physical and psychological condition of the elderly into consideration when developing collaborative applications like socially assistive robots. This work outlines HPLAN, an extension to the agent-framework JIAC V, that takes this requirement into account. HPLAN is strongly related to the conceptual model of dynamic planning components and integrates humans as avatars into a life cycle of planning, execution and learning.

1 Introduction

Following *H.H. Clark* [5, p. 3], joint human-agent activities can be defined as an extended set of actions that is executed by an ensemble of natural and artificial agents who are coordinating with each other [5, 16]. These agents coordinate to overcome their inherent limitations. As examples, consider agents with a sensory malfunction (perception level), humans with a disease like dementia (cognition level) or robots that are not able to overcome obstacles like stairs (execution level).

Planning procedures that account for joint human-agent activities are computed by Human-Aware Planning (HAP) components [4]. HAP is mainly required when the situation involves artificial and natural agents in the same environment, the actions of the artificial agents being planned and those of the natural agents being predicted only. One assumption of currently available human-aware planning components (e.g., [2–4, 14, 19]) is that whenever a human is predicted to fulfil a task, the human will provide results in a timely fashion. This assumption is questionable since the 'Quality of Behaviour' that a human is able to provide differs for each human. For instance, consider the activities of daily living [23] (ADL)—a measure for the self-sustainability of elderly people. Whether an elderly person is able to perform an ADL depends on the persons physical and psychological condition. Therefore it is necessary for planning agents to take such information into consideration. This work presents the first steps to use this kind

of information and to relax the mentioned assumption to a more general one. That is, whenever a human is predicted to fulfil a task, the human may perform the task or not and provide results either in time or delayed [1]. We introduce an extension to the agent-framework JIAC V [13, 17] named HPLAN, which enables the development of joint human-agent activities by providing three capabilities: (1) a generic link to several AI planners, (2) the use of additional information to influence the action selection of a planning process and (3) the integration of reinforcement learning techniques to adapt the additional information to the individual [7].

Indeed, the main contribution of this work is the presentation of an implementation, as the related work mainly presents conceptual frameworks. For example, *Kirsch et al.* [14] proposes a combination of TRANER and RoLL. TRANER is a planning system providing a library of reactive plans for autonomous household robots. RoLL is a robot programming language with a strong focus on machine learning. The combination of both enables to transform the available plans based on experience made during the execution. The work states that the strength of the system are not applied planning/learning techniques but the concept of combining two frameworks to facilitate joint human-agent activities. *Alami et al.* [2] propose to adjust the planning process to different types of humans using InteractionAgents each one providing information about an individual. The concept lacks details about the use of such information during the actual planning. *Cirillo et al.* [4] presents a more advanced solution combining activity recognition and the conceptual model of planning components. Other existing approaches [3, 19] plan without providing additional information about the human agents. Nevertheless, several authors emphasise to take such information into account [15, 19].

The remainder of this work is structured as follows. First, we will outline the bigger picture of our study using the already mentioned example of planning ADL, e.g. when developing socially assistive robots [22] in Section 2. In Section 3 we describe the approach combining available techniques to create a development environment for joint human-agent activities. Section 4 presents a technical evaluation using the Blocks World [11], where humans as additional actors suffering from weakness are introduced and cooperate with a robot to solve Blocks World problems. This scenario is far away from a real-world scenario and simulates a cooperative setting. Nevertheless, it was chosen to technically evaluate whether the design-decisions done are applicable to cooperative settings and to gain first experiences developing applications with HPLAN. Eventually, we conclude the work and give an outlook on future work in Section 5.

2 Motivation

To outline the objective of the work, imagine a socially assistive robot helping elderly people to stay independent at home. Such a robot should support the elderlies in the activities of daily living. Here older adults and robots cooperate to maintain the personal autonomy of the elderly. Yet, the aim of the cooperation

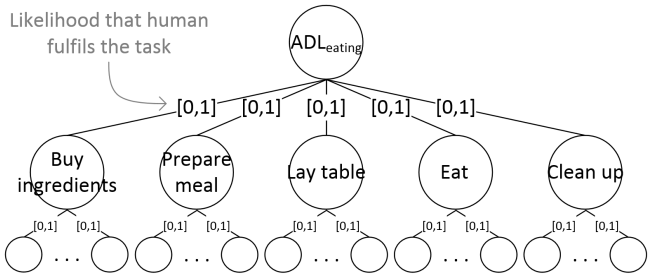

Fig. 1: Eating is one of the activities of daily living, visualised here as a hierarchical task network describing the decomposition of the complex task $\text{ADL}_{\text{eating}}$ to atomic actions (e.g. a possible leaf named pay for ingredients). Whether an elderly person is able to perform the task depends on its physical and psychological condition making the availability of such information necessary for agents planning procedures. The goal of our approach is to provide an estimate of the likelihood that an action is successfully executed by the human.

it not to leave all task to the artificial agents, but to help the human as much as necessary and as little as possible. Planning for this kind of joint human-agent activity requires knowledge about which agent—the artificial or the human—can perform which task and how likely the task is achieved. For this purpose the agent planning needs to predict the course of action of the human. Fig. 1 illustrates this objective for the ADL eating that comprises not only the actual consumption but also preparation and follow-up tasks.
The main objective of our work is to provide more information about the human—in terms of habits, abilities, personality and behaviour—and to use this information to sharpen the likelihood that the human can/will fulfil a task. For instance, the ability of a human to slice ingredients during the prepare meal task depends on the physical and psychological conditions of the human. These conditions can be influenced by diseases like Parkinson and its accompanying symptoms like tremble. As another example, weakness as a symptom of several diseases can diminish the ability to set up or clean up a table. To sharpen the estimated value of how good a human can fulfil a task, we want to investigate the use of additional information in combination with learning techniques. In particular, we do not aim to implement a new planning system but use existing ones as black boxes.

3 Approach

The goal of this work is to present a way to provide more information about humans to the planning process of joint human-agent activities. We want to accomplish this in terms of a more accurate cost estimate for specific capabilities. The costs are used to indicate the likelihood that a task will be performed, *i.e.*

lower cost indicates a higher likelihood and vice versa. We do not aim to develop a new planner, but to use existing solutions as far as possible. In consequence, our approach is to use the cost estimates to influence the action selection of a planning process, where the actual planning procedure is a black box. The idea is to integrate the costs into the planning process providing a—roughly speaking—dynamic heuristic about the possible course of action of a human user. *Sisbot et al.* [20] already showed the usefulness of this idea in the adjacent research field of Human-Aware Navigation. The authors used the A* search algorithm [12] for the motion planning of robots. Such robots should avoid to approach humans from behind during the motion. To accomplish this the authors attached higher cost to actions in the back of humans and thus influenced the path-finding without changing the algorithm.

3.1 Agent-Model Construction

To transfer this idea to a collaborative setting, we represent each human as part of the agent-system similar to the concept of InterActionAgents presented by *Alami et al.* [2]. In this work avatars named actor agents each represent a human or an artificial agent. Each actor agent representing a human provides information about the capabilities and the personality of the human. In a formal way this can be expressed using the following agent-model for an actor agent a_h:

$$\{A_{a_h}, P_{a_h}, cost : A_{a_h} \times P_{a_h} \to \mathbb{R}\}.$$

Here, $A_{a_h} \subseteq A$ is the set of capabilities (actions) that a human is able to provide. In our example, A_{a_h} would include actions necessary for the activities of daily living. The behaviour of a human is represented by the set of personality $P_{a_h} \subseteq P$, where each $p \in P$ represents a personality trait with range $[0, 1]$. This abstraction serves as a wild-card for a specific type of information. For our example, this might be a psychological trait from a theory like the Five Factor Model [18] or information about a disease. The agent-model is completed with the relation *cost* between actions and personality. This relation is used to dynamically assign costs in terms of a real number to each action, which will later be used to generate plans with minimised costs.

3.2 Planning for Joint-Human Agent Activities

A system suitable for a planning process for joint human-agent activities needs to create a plan, execute it, learn from the execution and start over. Therefore it must be able to determine the current state of the environment, to detect failure and to replan if necessary. Furthermore, experiences generated from the execution of actions must be used to improve the task delegation process.

Concept Fig. 2 illustrates the architecture of our approach and visualises the relationship to the conceptual model for dynamic planning systems introduced by *Ghallab et al.* [10, p. 9]. Here, the controller handles the execution of plans

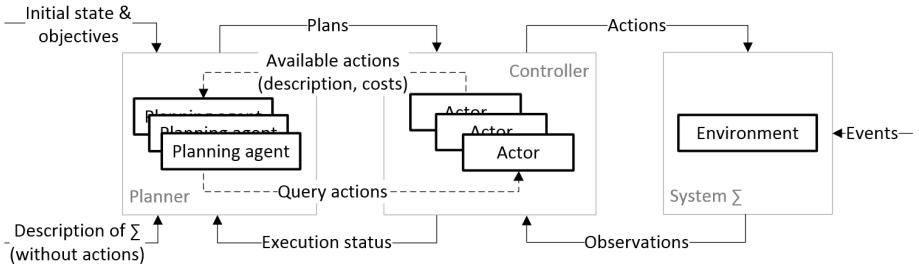

Fig. 2: High-level architecture of HPLAN visualised as part of the conceptual model for dynamic planning components (greyed out) [10, p. 9]. The planning capabilities are provided by planning agents, which can be actors as well. The initial domain description encloses no action descriptions as the available actions are only provided by the actors at runtime. The process starts with a new objective triggering an agent with planning capabilities to query all available actors. The actors then provide an action description and a cost estimate for each action that they want to offer for the planning process.

generated by the planner based on an initial state and a set of goals that are provided by an external source. The controller executes actions, processes observations from the environment and informs the planner about the plan execution status. In our approach, a controller contains a set of actors, each representing a human or an artificial agent that is capable of manipulating the environment. The planning process for a new objective starts by querying all actor agents about their available actions and the associated cost predictions to generate a full domain description. The query is executed by an agent with planning capabilities, which can be an actor agent or an agent solely responsible for planning processes. The queried information is used to generate a plan in which each task is delegated to the most capable agent. During the execution, the observations generated in the environment are evaluated by the associated actor. If a failure occurs, it is reported back to the planner to trigger replanning. Furthermore, the actor agents representing a human learn from each execution experience and adapt their cost predictions accordingly, therefore completing the life cycle of planning, executing and learning.

Implementation The described structure was implemented as an extension module for the agent-framework JIAC V [13, 17] (Java Intelligent Agent Componentware – Version V). We extended the action annotation process used in the agent-framework with the ability to annotate human-action descriptions. In JIAC V the expose annotation is used to declare an agent's actions. Listing 1.1 shows an annotation for an action named *sliceIngredients*. At runtime all relevant information is extracted from the annotation and its attributes. Here, the *name* of an action is used to register it in the dictionary of each agent platform (the dictionary is a yellow-page service). The *scope* of the action is used to control

```
@Expose(
  name = ADL_EATING_PREPARE_SLICE,
  scope = ActionScope.GLOBAL,
  actor = ActorType.HUMAN,
  descr = ADL_EATING_PREPARE_SLICE_DESCR
  cost = ADL_EATING_PREPARE_SLICE_COST
)
public void sliceIngredients(){
  // Implementation of user interaction
}
```

Listing 1.1: Example of annotating actions when developing actors that represent humans.

its visibility. It controls whether the action is visible to all existing agents, to the agents on a single platform or only to the agent owning such action. The *actor* defines if an action is provided by an artificial or natural agent. The *descr* provides the action description in the selected planning language. The *cost* attribute holds the current cost estimate for this action by this actor. This estimate is automatically embedded into the action description when the planning agents queries the multi-agent system about the available actions.
As indicated in the listing, developers have to provide a description for an action and the way the actor interacts with the human user when the action is used. Developers are not required to implement the action's logic, as the action will be executed by humans. For the actual planning, we implemented a planning module using the planning library Planning4J.[1] This approach enables planning with various AI planners, even if they are written in other programming languages than Java. Actions are described using the planning language PDDL [9], which was objectified to ease the manipulation of the associated action costs and to support reusability. We use the concept of numerical fluents first introduced in PDDL2.1 [8] to assign costs to actions. Furthermore, we use the minimisation plan-metric—also first introduced with PDDL2.1—for the quantitative directive of plan creation.

3.3 Stateless Q-learning to estimate Costs

We apply stateless Q-learning [6] in order to estimate the expected costs of executing an action according to the personality traits of an agent. We drop the state dependency as the goal of this work is to learn the ability of an agent to fulfil a specific task, not to learn the utility of an action in a specific state of the environment, which will be done in the future.
By definition, a learning agent interacts with its environment by performing an action a at time t. In return, the agent receives a reward $r_t(a)$ and iteratively improves its estimate $Q_t(a)$ of the expected reward for each action a. In other words the agent builds an estimate of the expected costs of executing an action

[1] For more information about Planning4J and the supported AI planning solutions the interested reader is refereed to `https://code.google.com/p/planning4j`.

a. This estimate is iteratively updated using the following equation, known as the Q-learning update rule. Here, the parameter α with range $[0, 1]$ denotes the learning rate, helping to control the influence of new experiences to the current cost estimate:

$$Q_{t+1}(a) \leftarrow Q_t(a) + \alpha \left(r_t(a) - Q_t(a) \right).$$

We choose the Q-learning update rule because the sample-average method would not react fast enough to changing capabilities of humans in the long run (e.g., if a human gets tired of performing a repetitive task). To learn from feedback, the effects of actions have to be evaluated by some criteria according to the type of the reward signal. Such a reward signal can be of qualitative (e.g., in terms of 'failure' and 'success') or quantitative nature (e.g., in terms of time steps required to execute an action). As both signal types require different computation and the interpretation of the reward signal is domain dependent, we developed different interfaces to encapsulate the actual implementation. Fig. 3 shows a more detailed view of an actor agent and introduces some of the interfaces, which are provided for developers.

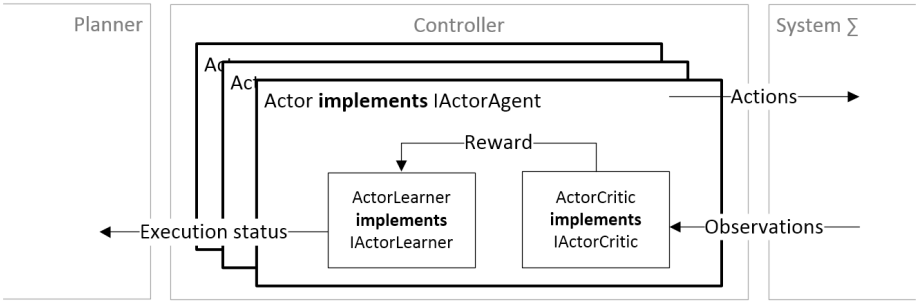

Fig. 3: A more detailed view of an actor agent. Each JIAC V agent consist of several components named AgentBeans which encapsulate functionalities. An actor representing a human is equipped with at least two AgentBeans, one named ActorCritic and one named ActorLearner. The ActorCritic evaluates the observations and is responsible for generating the reward signal. In other words, the ActorCritic preprocesses observations, which can for example be derived from sensor signals, to generate a computable reward signal such as 'failure' or 'success'. The ActorLearner then uses these reward signals to adjust the cost of the associated actions using machine learning techniques. In its current implementation the ActorLearner uses stateless Q-learning.

The above-mentioned learning procedure is applied by the available actors each time they manipulate the environment. Agents with planning capabilities then use this information to produce plans with minimal overall costs. To ensure that the approach is able to reach minimised plan costs, we applied the ϵ-greedy policy

as the action selection strategy (if not otherwise stated we use an exploration rate of $\epsilon = 0.1$) [21]. This guarantees that $Q_t(a)$ converges to $Q^*(a)$ for $t \to \infty$, where $Q^*(a)$ is the mean reward received when a is executed [21].

4 Case Study

We used a classical planning problem—the Blocks World [11]—to evaluate the presented approach. The Blocks World in our evaluation scenario contains two types of effectors, namely robots and humans. Each can move blocks, but the efficiency of humans is higher on average. Humans suffer from weakness and in consequence have the potential to make errors for tasks that involve boxes that are more than one level of the ground. Related to our examples, this might be the task to carry dishes from the table to the wall cupboard or vice versa. We introduce two types of errors: Failure at moving a block (denoted as external factor ext_c) and the timely execution of moving a block (denoted as external factor ext_r). The external factors ($ext_{c,r}$) serve as hidden properties not accessible to the planning system and not known to the human-agent representing a human. In consequence, the human-agent must observe the environment during the action executions by its associated human. To represent this information, we use the hidden properties as personality traits for the actor agents representing the human. The goal is to determine the helpfulness of a human being to reach a given goal with $P_{HELP} = \{p_c, p_r\}$. Cooperation (p_c) measures a human's ability to fulfil a task. It indicates the likeliness that the human succeeds or fails to execute a given task. Reliability (p_r) measures a human's ability to provide results either in time or delayed. It indicates the likeliness that a task will be processed in time and the expected time delay.

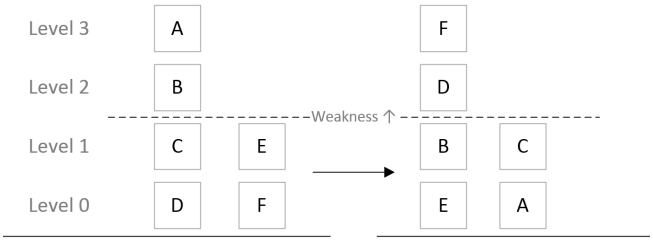

Fig. 4: Evaluation scenario visualising the initial state (left) and the goal state (right). Each action that is executed by a human actor more than one level of the table triggers weakness.

Fig. 4 shows one evaluation scenario. Given this scenario, we are able to manually calculate minimal plan costs for different cooperation and reliability values. To test whether the system adapts to dynamic external factors during runtime, we

change the human behaviour after $n = 50$ solutions from $ext_c = 0.3, ext_r = 1.6$ to $ext_c = 0, ext_r = 1$, simulating a human that does not suffer from weakness and requires one time step to execute each action. After 50 experiences at $n = 101$ we restore the previously used external factors, simulating a human that fails in 30% of all weakness triggering tasks and requires 1.6 time steps to execute such actions. The change in behaviour enables us to observe the ability of the system to adapt a model that was already learned.

Given these requirements, we implemented an actor agent that is able to process observations to determine the helpfulness of a human. Here, the ActorCritic distinguish a qualitative reward signal in terms of 'failure' and 'success', which is used to learn p_c, and a quantitative reward signal in terms of time steps required to execute an action, which is used to learn p_r. As both require different computation, the reward signal is processed using the following twofold equation:

$$r_t(a) \leftarrow \begin{cases} \rho \times c(a) & \text{if 'failure'} \\ \Delta_t(a) & \text{otherwise} \end{cases} .$$

Here, the parameter ρ is a constant factor to punish the execution of an action a if the execution has failed, whereas $c(a) \leftarrow Q_{t_0}(a)$ is the initial cost estimate for action a provided by the developer. The execution time of an action a is denoted as $\Delta_t(a)$.

To determine whether the system improves the cost effectiveness of solving a problem, we use the average cost $C_n^m = \frac{\sum_{i=0}^{m} c_{i,n}}{m}$ to solve the problem after n previous experiences averaged over m rounds. Each experience solves one instance of the problem, including necessary replannings. The use of this average value removes statistical variations introduced by ext_c and ext_r for large numbers of m. If not explicitly stated, we will use $n = 1 \ldots 150, m = 100$ for the experiments. Furthermore—in a real world scenario—humans would expect the planner to produce legible behaviour and therefore consider if a human feels safe and comfortable [15]. Using the cost-progression is not suitable to show this, as a human has a different point of view on what an optimal plan is. Humans in our scenario would prefer a plan that delegates as few weakness affected tasks to them as possible if they are suffering from weakness. Humans suffering from weakness might also not want to be frequently asked to execute tasks they are not able to perform.

Fig. 5 illustrates the number of tasks inducing weakness that are assigned to the human agent and the robot. The number of weakness affected tasks that the human has to execute decreases significantly for the two stages in which the human suffers from weakness. The graph confirms that tasks that are not executed by the human are executed by the robot instead. At the beginning of stage_1 more weakness affected tasks are executed by the human than required to solve the problem. This is due to the fact that a number of failures occur and the tasks therefore have to be executed multiple times (underpinned by the number of replannings). As the planner delegates all tasks to the human at first, the human executes up to 5.4 tasks to solve a problem. Note that this behaviour was expected as in the initial action description the humans performance were

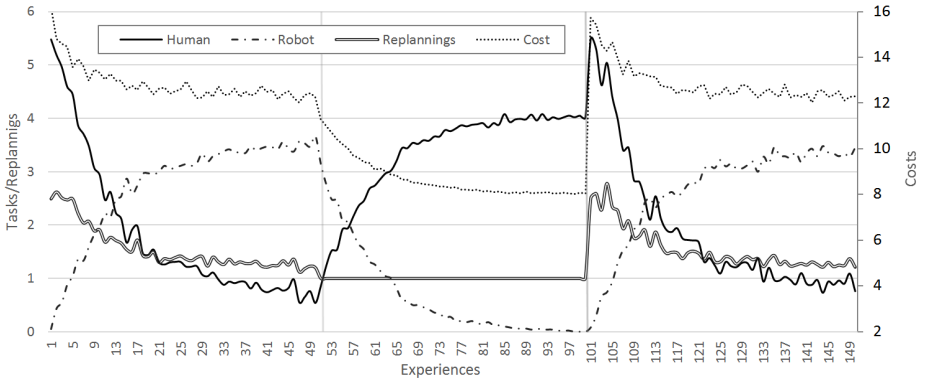

Fig. 5: Diagram shows the number of weakness triggering tasks executed by the human agent, number of task executed by the robot, number of replannings done during the simulation and the overall cost-progression ($\alpha = 0.1$). The optimal plan costs are $stage_{1,3} = 12, stage_2 = 8$. In $stage_2$ the human does not suffer from weakness, to test whether the system adapts to dynamically changing behaviour. For evaluation purposes we used scenarios small enough to enable cost calculation manually.

assumed higher on average (2 : 1). This was done to ensure that the planner tries to assign actions to the human actor frequently. At the end of stage_2 the planner is correctly confident that the human is not suffering from weakness and the human is delegated all four weakness affected tasks that are required to solve the problem. At the transition point between stage_2 and stage_3 the planner is still confident that the human does not suffer from weakness. The human then changes its internal model and the system again has to replan multiple times until it adapts to the failure rate of the human agent. This creates a peak in the number of weakness affected tasks executed to solve the problem. Associated with this observation the costs drop to the near optimum during all three stages.

To show that the system learns problem independent, we tested the use of an already adapted model to solve other problems. To show this, we replace the problem with a different one after $n = 30$. The time-optimal solution plan for this problem takes 8 time steps. If the system does indeed learn problem independent behaviour, we would expect the system to perform efficiently on the second problem without adaptation. Fig. 6 shows, that the system reaches an near time-optimal solution on the second problem without additional adaptation (also underpinned by the number of replannings). We can therefore conclude that the system does not simply learn problem specific behaviour but indeed learns the hidden properties of the human agents. Furthermore, we can conclude that the additional information forwarded in terms of a more accurate cost estimate influence the action selection of the planning process in a positive manner.

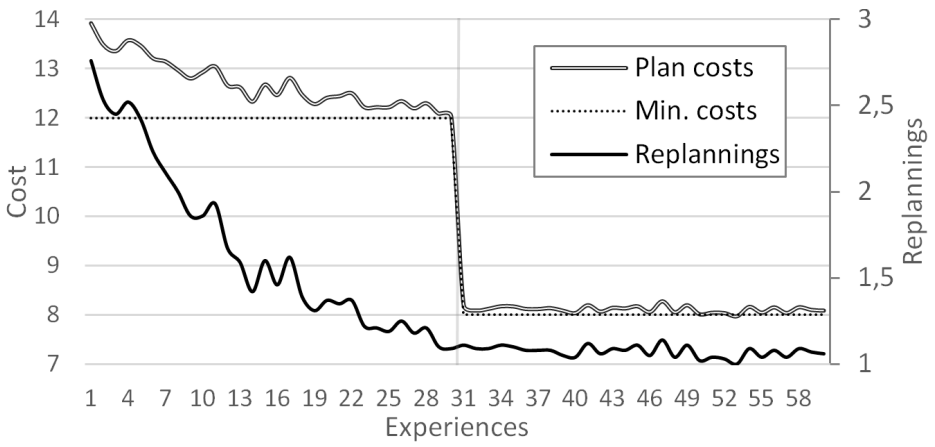

Fig. 6: Cost progression and number of replannings when changing the problem during runtime ($n = 1 \ldots 60, m = 100, \epsilon = 0.01$).

5 Conclusion

We presented an agent-based architecture named HPLAN that facilitates the development of joint human-agent activities. HPLAN is strongly related to the conceptual model of planning and implements the life cycle of planning, executing and learning. The related work shows that contemporary solutions lack in terms of learning and adapting to humans and in consequence sacrifice potential in terms of planning efficiency and the generation of legible behaviour. The presented technical evaluation shows that our approach is able to reach near optimal plan cost after a short number of experiences. The approach also reduces the number of non-optimal action assignments to humans. The evaluation results indicate, that the concept of forwarding additional information to the planning component is promising in terms of a more accurate cost estimate. Nevertheless, the case-study is just a technical evaluation of the system emphasising that HPLAN facilitates the implementation of joint human-agent activities. In future work it will be interesting to see if using one Q-learner for each information is applicable for real-world applications. Furthermore, currently each additional information influences the cost of each action, leaving the actual context out of consideration. It might be necessary to find a more fine-granular way since the 'Quality of Behaviour' a human is able to provide not only differs for each human but also differs for several contexts.

References

1. Ahrndt, S.: Improving human-aware planning. In: Klusch, M., Thimm, M., Paprzycki, M. (eds.) Multiagent System Technologies, pp. 400–403. No. 8076 in Lecture Notes in Artificial Intelligence, Springer (2013)

2. Alami, R., Clodic, A., Montreuil, V., Sisbot, E.A., Chatila, R.: Task planning for human-robot interaction. In: Bailly, G., Crowley, J.L., Privat, G. (eds.) Proc. of the sOc-EUSAI2005. pp. 81–85. ACM Press (2005)
3. Alili, S., Warnier, M., Ali, M., Alami, R.: Planning and plan-execution for human-robot cooperative task achievement. In: Proc. of the 19th ICAPS. pp. 1–6 (2009)
4. Cirillo, M., Karlsson, L., Saffiotti, A.: Human-aware task planning: An application to mobile robots. ACM Trans. Intell. Syst. Technol. 1(2), 1–26 (2010)
5. Clark, H.H.: Using Language. Cambridge Univ. Press (1996)
6. Claus, C., Boutilier, C.: Thy dynamics of reinforcement learning in cooperative multiagent systems. In: Proc. of the 15th AAAI. pp. 746–752 (1998)
7. Ebert, P.: Improving Human-Aware Planning through Reinforcement Learning – A Multi-Agent Based Approach. Master's thesis, TU Berlin (2013)
8. Fox, M., Long, D.: PDDL2.1: An extension to PDDL for expressing temporal planning domains. Artifical Intelligence Research 20, 61–124 (2003)
9. Ghallab, M., Howe, A., Knoblock, C., et al.: PDDL – The Planning Domain Definition Language. Yale Center for Computational Vision and Control (1998)
10. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann (2004)
11. Gupta, N., Nau, D.S.: On the complexity of blocks-world planning. Artifical Intelligence 56(2–3), 223–254 (August 1992)
12. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. Systems Science and Cybernetics, IEEE Transactions on 4(2), 100–107 (July 1968)
13. Hirsch, B., Konnerth, T., Heßler, A.: Merging agents and services – the JIAC agent platform. In: Bordini, R.H., Dastani, M., Dix, J., Amal, E.F.S. (eds.) Multi-Agent Programming: Languages, Tools and Applications, pp. 159–185. Springer (2009)
14. Kirsch, A., Kruse, T., Mösenlechner, L.: An integrated planning and learning framework for human-robot interaction. In: Proc. of the 19th ICAPS. pp. 1–6 (2009)
15. Kirsch, A., Kruse, T., Sisbot, E.A., et al.: Plan-based control of joint human-robot activities. KI – Künstliche Intelligenz 24(3), 223–231 (2010)
16. Klein, G., Woods, D.D., Bradshaw, J.M., Hoffmann, R.R., Feltovich, P.J.: Ten challenges for making automation a 'team player' in joint human-agent activity. Human-Centered Computing 19(6), 91–95 (2004)
17. Lützenberger, M., Küster, T., Konnerth, T., et al.: JIAC V –A MAS framework for industrial applications (extended abstract). In: Ito, T., Jonker, C., Gini, M., Shehory, O. (eds.) Proc. of the 12th AAMAS. pp. 1189–1190 (2013)
18. McCrea, R.R., John, O.P.: An introduction to the five-factor model and its applications. Personality 60(2), 175–215 (1992)
19. Montreuil, V., Clodic, A., Alami, R.: Planning human centered robot activities. In: IEEE SMC. pp. 2618 –2623 (2007)
20. Sisbot, E.A., Marin-Urias, L.F., Alami, R., Simeon, T.: A human aware mobile robot motion planner. IEEE Transactions on Robotics 23(5), 874–883 (2007)
21. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. Adaptive Computation and Machine Learning, MIT Press (May 1998)
22. Tapus, A., Matarić, M.J., Scassellati, B.: The grand challenges in socially assistive robotics. IEEE Robotics and Automation Magazin 14(1), 35–42 (2007)
23. Wiener, J.M., Hanley, R.J., Clark, R., Nostrand, J.F.V.: Measuring the activities of daily living: Comparison across national surveys. Tech. rep., U.S. Department of Health and Human Services (1990), `http://aspe.hhs.gov/daltcp/reports/meacmpes.pdf`, last access: 2014-02-25