EPiC Series in Computing

Volume 41, 2016, Pages 187–199

GCAI 2016. 2nd Global Conference on Artificial Intelligence

A Short-Term Memory for Deliberative Agents in Everyday Environments

Ivo Chichkov[1] and Alexandra Kirsch[2]

[1] Eberhard Karls Universität Tübingen Tübingen, Germany
ivo.chichkov@student.uni-tuebingen.de
[2] Eberhard Karls Universität Tübingen Tübingen, Germany
alexandra.kirsch@uni-tuebingen.de

Abstract

Humans have the impressive capability to efficiently find near-optimal solutions to complex, multi-step problems. AI planning can model such problems well, but is inefficient for realistic problems. We propose to use AI planning in combination with a short-term memory, inspired by models of human short-term memory, to structure real-world problem domains and make the planning process more efficient, while still producing satisficing solutions. We evaluate the method in the domain of a household robot.

1 Introduction

Machines still lack the ability to generate solutions for everyday problems that fulfill the standard of human problem solving, be it personal schedule organizers, route guidance systems, or autonomous robots. All of these applications need some degree of goal-directed reasoning, as it is available in existing reasoning frameworks such as AI planning, but in a typical household we find hundreds of objects like shoes, toothbrushes, pots, plants, etc., allowing for a large number of actions. While current AI planners can solve problems to a certain size (near-)optimally, the state space implied by a realistic household is beyond anything that is possible today or in the near future.

Yet humans are well capable of dealing with household tasks, even though their computational resources are restricted, possibly more than those of computers, for example in terms of working memory capacity.

In the environments humans usually work and live in, the goal is not optimality, but rather efficiency and flexibility. One striking restriction in human computational resources is the limited capacity of working memory. We conjecture that this phenomenon is not just a curse, but may (partly) be a blessing. Assuming that humans can only access the knowledge in working memory, any search they may perform is naturally restricted to a small search space.

In this paper, we explore how available, well-developed methods of AI planning can be combined with the concept of short-term (or working) memory from cognitive science to reduce the search space while keeping an acceptable level of plan quality. This goal is fundamentally

different from the goal of developing optimal or provably near-optimal planning algorithms. While for some domains optimality is a crucial factor, in everyday domains, such as a household, it is not (as we can see in humans). We introduce a model called PDDLMemory that integrates a cognitively inspired memory model with standard PDDL planning. We illustrate PDDLMemory in a household robot domain, showing how planning times can be significantly reduced with a short-term memory module, while still producing adequate solutions to planning problems, even with a simple memory management strategy.

2 Memory Models

Memory has always been a widely discussed topic in the study of human cognition. Experimental results have stimulated the formulation of different models of memory. The following is a brief overview of the most influential ideas from psychology.

2.1 Organization of Memory

Some models regard memory organization as a collection of separate storage components. In the Multi-Store Model of Atkinson and Shiffrin [4], [11, p.6], memory consists of the sensory register, the short-term store, and the long-term store. Information first enters memory through the sensory register. Attentional processes determine which pieces of information will enter the short-term store. Rehearsal processes are responsible for maintaining information in the short-term store. Finally, some piece of information maintained in the short-term store gets transferred to the long-term store.

The idea of multiple stores has been further developed by Baddeley and Hitch [5]. In their Multicomponent Model, the short-term store (or short-term memory) is replaced by a more complex component called working memory. The term working memory was introduced in order to emphasize the fact that this part of memory is not merely a passive storage component. Baddeley and Hitch's working memory consists of additional components such as the phonological loop and the visuo-spatial sketchpad. These are responsible for the temporary storage and processing of auditory and visual information.

The models presented so far assume a strict separation between the different parts of memory; memory items are transferred from one component to another.

A different type of memory organization assumes a fuzzy boundary between the different components like short-term or long-term memory. The Embedded-Processing-Model [8] divides memory into three layers: the long-term store, the activated memory, and the focus of attention. The items are not transferred between different parts of memory. Instead, a part of long-term memory resides in an activated state. In contrast to Multi-Store models, there is no limitation on the number of items that can be activated simultaneously. However, activated memory is subject to time limitations — if activation is not maintained, items reside in activated memory for about 10–30 seconds. The innermost layer of the model is the focus of attention. In Cowan's model, the focus of attention is limited to three to four items that can be attended to simultaneously.

2.2 Retrieval and Forgetting

Retrieval can be described as the process by which represented information from the inactive portion of memory (i.e., the long-term store) becomes activated. Incoming stimuli play the role of *cues*. These are pieces of information that are associated with knowledge in the inactive

part of memory, also called *targets*. As soon as the cue stimulus is observed, retrieval processes are initiated that lead to the activation of the associated targets. Every stimulus that we are exposed to may lead to the conscious or subconscious activation of inactive memories [8].

Forgetting can be understood as a time-based or as an event-based mechanism [23]. Another explanation for the lost access to memories is the displacement or overshadowing of items by other items in memory [10, p. 271].

2.3 Units of Memory and Memory Capacity

The units of information stored in memory are sometimes called memory *items*.

Miller [22] suggested that there is a limit on the number of items that can be stored in short-term memory and that this limit is 7 ± 2 items. Related information can be combined into information units (chunks), allowing for the storage of larger pieces of information.

More recent approaches [1] have suggested that there is no fixed limit on the number of items, but there is an "information limit on working memory, which would predict a trade-off between the number of items stored and the fidelity with which each item is stored". Basically, this means that more items can be stored if some of them are encoded in less detail.

2.4 Computational Models of Memory

Computational models of memory have been studied in the context of cognitive architectures [17]. These architectures are motivated by modeling human cognitive abilities and thus contain very detailed memory models.

For example, ACT-R [2] has different (long-term) memory stores for visual processing, goals, and declarative knowledge. Short-term memory is modeled by buffers for each of these long-term stores [17]. Similar to our model, ACT-R reduces the space of possible actions by matching production rules to the contents in the short-term memory buffers.

Soar [9] differentiates between episodic and semantic memory. It comprises a working memory, where (de-)activation of items is driven by the Soar decision process and includes different cognitive mechanisms such as temporal decay [9].

Icarus [18] uses a hierarchical, ontology-like long-term memory structure, where the assignment of objects to classes is probabilistic. Whereas ACT-R and Soar use production rules as the basic decision-making process, Icarus works with goals and plans. It uses a hierarchical plan structure with transformational planning. Thus, it comes closest to the decision-making method of PDDLMemory, but does not fit directly into standard AI planning based on PDDL. Icarus stores the agent's beliefs in short-term memory, which trigger concepts in the long-term memory structure. The reasoning can thus use any object in long-term memory if it is accessible via the activation of a currently held belief.

All these architectures support sophisticated models of long- and short-term memory, closely modeling human cognitive processes. In contrast, PDDLMemory is a lightweight extension of AI planning, making use of existing tools. In this context, we can specifically examine the effect of a short-term memory without considering interactions with other cognitive modules.

3 The Planning Domain Definition Language

In general, a planning problem is defined by a transition system $\langle \mathcal{S}, \mathcal{O}, \gamma \rangle$, with states $\mathcal{S}$, operators $\mathcal{O}$, and a state transition function $\gamma : \mathcal{S} \times \mathcal{O} \to \mathcal{S}$. Given an initial state s_0 and a

```
(define (domain apartment)
  (:requirements :strips :typing)
  (:types Location MovableObject Person Room - Object
          Window Floor Door Lightswitch Lamp - Location
          Lightbulb CleaningProducts - MovableObject)
  (:predicates (own-location ?l - Location)         ; Own Location
               (is-at ?l - Location ?o - Object)    ; Location of Object
               (is-in ?r - Room ?l - Location)      ; Location is in Room
               (is-dirty-loc ?l - Location)         ; Location is dirty
               (is-dirty-mobj ?mo - MovableObject) ; Object is dirty
               (is-attached ?mo - MovableObject)
; Object is not movable
               (is-intact ?lb - Lightbulb)          ; Lightbulb is intact
               (is-working ?l - Lamp))              ; Lamp is working
  ...)
```

Figure 1: Type and predicate definitions of the household domain.

specification of a set of goal states S_g, the task is to find a plan $\Pi = (o_1, o_2, \ldots, o_n), o_i \in \mathcal{O}$, so that the successive application of $o_1, \ldots o_n$ starting from s_0 results in a state $s_f \in S_g$.

The Planning Domain Definition Language PDDL [21] is the standard language for state-of-the-art planners. It has been developed along with the International Planning Competition and has been extended to incorporate sophisticated representations beyond classical planning such as time and uncertainty. In this paper, we only use the basic deterministic version of PDDL.

In PDDL, states are represented by predicates $\mathcal{P}$ over typed object variables $\mathcal{V}$. Operators are defined abstractly as actions $\mathcal{A}$ with a type signature, and state transition function γ. Actions are patterns of operators that are instantiated by the planner. PDDL differentiates between *domain definitions*, containing the specification of types, predicates and actions, and *problem definitions*, defined by a list of available objects, the initial state and a set of goal predicates.

The following example of a household environment illustrates the components of PDDL and will be used throughout the paper.

Figure 1 shows the definition of types and predicates from our PDDL domain definition file. We further define the following actions:

- go (to location in room);
- move (object to location in room);
- clean (location with cleaning product);
- change-lightbulb (in lamp with lightbulb);

For the planning problem, we define an apartment with different rooms as shown in Figure 2(a). Each room has a set of distinctive locations and objects. Table 2(b) gives an overview of the size of this problem definition.

In this domain, the robot has a set of goals such as preparing tea utensils, putting away shoes, or cleaning the bathroom. In our trials, we assume that the robot has a given list of chores it has to fulfill over the day, modeled as a conjunction of individual goals. But the system is designed with the possibility in mind that a user could add or remove goals at any time.

4 Approach

We first sketch the mechanisms that differentiate planning with PDDLMemory from classical AI planning, matching the goals described in Section 1. First, we transform the PDDL problem description to basic units of PDDLMemory — items and chunks. The initial decomposition into chunks is similar to the decomposition of a problem into subproblems [25] in the sense that both approaches aim for a problem representation that corresponds to a partition of the original problem. However, the subproblems defined by chunks are not necessarily independent. In our implementation, solving a subproblem may involve accessing information from different chunks.

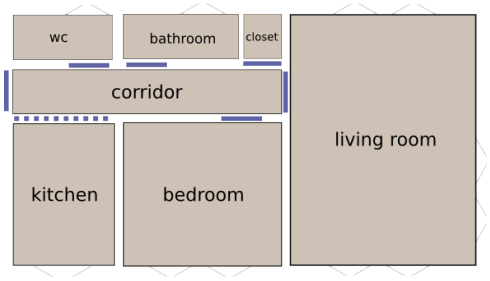
Second, planning with PDDLMemory proceeds in iterations, each solving a different simplified version of the overall problem, depending on the activated items in memory. The fact that some of the simplified problems cannot be solved is taken into account by the iterative procedure and the assumption that the environment is dynamic, so that changes are rather beneficial to this approach, enabling the agent to use new information to fulfill its goals.

4.1 The PDDLMemory Module

PDDLMemory follows roughly the embedded processing model of Cowan [8], but with only two layers of active and inactive memory. We chose this model, because we embrace the idea that the items in short-term store are pointers (and not copies) to items in long-term store and that the model imposes no general restriction on the size of the activated portion of memory (whereas multi-component models rely on the 7 ± 2 rule), leaving the capacity as an open parameter. Even though in this paper we work with a fixed memory size, the model allows further research into the optimal capacity of the active memory.

Following the literature [6], we call basic units of knowledge that can be stored in memory *items*. Several items may be combined into a *chunk*. The basic units of information in PDDL are facts and goal predicates. Thus, an item in PDDLMemory corresponds to a single PDDL fact or goal predicate. A set of facts (PDDL predicates) and a set of goals may constitute a chunk $c = \langle P_c, G_c \rangle$. Actions are currently assumed to be known universally, but they could be modeled as memory items as well.

The PDDL domain definition is considered as background knowledge, independent from the knowledge in active memory, whereas the initial state and goal state of a problem are represented by the current content of active memory. Knowledge can be activated or deactivated at the level of chunks.



(a) Map of apartment modeled in PDDL.

Room	# locations	# objects
bathroom	5	9
WC	2	8
Closet	1	5
Kitchen	14	17
Corridor	3	7
Bedroom	6	12
Living room	3	19

(b) Number of defined locations and objects per room.

Figure 2: The household domain.

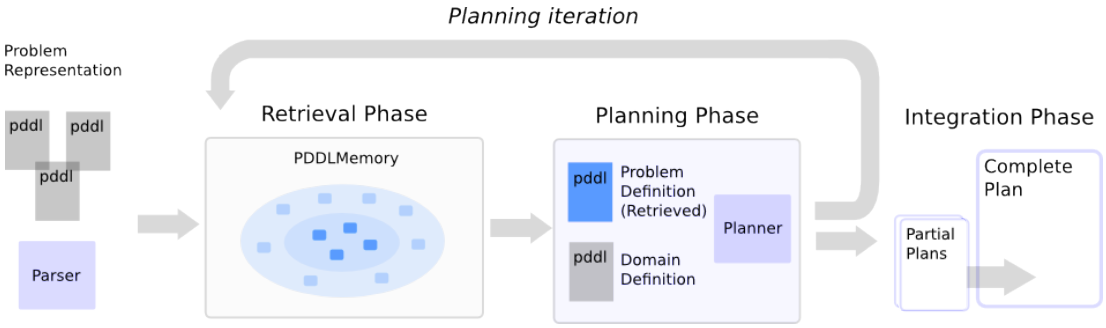


Figure 3: Overview of planning approach with PDDLMemory.

4.2 Implementation of Chunks as Place Nodes

In our household domain we chose to model chunks according to *places*. Psychological studies of human memory show that the environmental context plays an important role for storage and retrieval [11, p.176]. Places do not necessarily have to represent a geographical entity. They can include geographical information such as landmarks and directions, but can also be associated with other sensory cues or actions that may be carried out at that place [20].

We have modeled places corresponding to the rooms in our environment. By defining connections between those places, we derive a topological map, a representation that has also been assumed to be used by humans and animals [20].

We define a *PlaceNode* as a subclass of a Chunk, representing a specific room in the modeled apartment. It contains all the facts relevant to this room as well as a set of goals that affect primarily this specific room.

4.3 The PDDLMemory Iteration Loop

Planning with PDDLMemory proceeds in three phases: a retrieval phase, a planning phase, and a plan integration phase (Figure 3).

As input PDDLMemory needs a domain and problem description, which is introduced in the following definition.

Definition: PDDLMemory domain and problem descripion
A *PDDLMemory domain and problem description* is given by a 5-tuple $\langle D, M_i, M_a, \psi, \bar{c} \rangle$ with the following components:

- D is a traditional *PDDL domain description*.
- $M_i = \{c_1 \dots c_n\}$ is the *set of inactive chunks* with $c_i = \langle P_{c_i}, G_{c_i} \rangle$. Every chunk is composed of a set of PDDL fact predicates, P_{c_i}, and a set of PDDL goal predicates, G_{c_i}.
- $M_a = \{\bar{c}\}$ is the *set of active chunks* constituting short-term memory. Initially, it contains the universally known chunk $\bar{c}$.
- An *activation function* ψ.
- A *universally known chunk* $\bar{c}$.

The domain definition is universally known, independent from the current content of the activated memory. The same holds for some parts of the problem, e.g., the layout of the

apartment, which we model by a universally known chunk $\bar{c}$. All chunks except $\bar{c}$ are stored a priori in the inactive memory M_i.

PDDLMemory is a Python program that parses the fact and goal definition files, and combines them internally into chunk objects (mapping the directory structure to chunks). A chunk object has a state — active or inactive. For the *retrieval* phase, all chunks are first reset to an inactive state, and then a number of chunks corresponding to the capacity of short-term memory are marked as active. The strategy for activating chunks is described in Section 4.4.

In the *planning phase*, PDDLMemory transforms the activation pattern back to PDDL files by copying the facts and goal predicates of all activated chunks to a PDDL problem definition generated on the fly. This problem definition is passed to an AI planner, thus only providing globally known information and the knowledge in active memory.

If the planning is successful, the resulting plan π can directly be executed. However, in our experiments, where we compare PDDLMemory to planning with complete knowledge, we store the plan as a partial plan. When k partial plans have been found to fulfill all goals, the partial plans π_i are combined into a complete plan by concatenation, thus $\Pi = \pi_1 \circ \pi_2 \cdots \circ \pi_k$ (*plan integration phase*).

4.4 Memory Management

Memory management in our model favors the integration with deliberative architectures in realistic everyday environments.

In a realistic environment, a robot moves through the world when trying to achieve its goals. By moving through the environment, a robot senses the world and can use the newly acquired sensor information to activate memory items. Recall that items in our model are clustered into chunks, where each chunk is a PlaceNode representing a room in the appartment.

Every time a robot enters a room, the corresponding PlaceNode, i.e., the robot's knowledge of that room, is activated. The robot can use this knowledge during the planning phase to solve a task in that room. There are, however, more complex *multi-chunk* goals, which need knowledge from several chunks. For example, when the robot has to clean the bathroom, but has no knowledge about where to find the cleaning products, it can either pursue some other task or randomly explore its environment until at some point it enters the closet. This would activate its knowledge about cleaning products and remind it of the previously suspended goal.

Due to short-term memory constraints mentioned in Section 4.1 the number of chunks that can be activated simultaneously is limited. Thus, the contents of short-term memory has to be managed by processes of forgetting and activation.

The forgetting process in PDDLMemory is binary – chunks are either activated or they become part of the inactive portion of the long-term store. Unlike some cognitive architectures, we do not implement a time-based decay mechanism. It should be noted that as a rule of thumb we *forget everything* at the end of *every* main iteration. As a consequence, the process of activation plays an important role in our model.

To activate chunks, we apply the activation function Ψ to the set of available chunks M_i stored in inactive memory. Thus, the activated memory of capacity n can be represented by a set of chunks $M_a = \{\bar{c}, c_1, \cdots, c_m\}$, where $M_a \leftarrow \Psi(M_i), M_i \leftarrow M_i \setminus M_a$. In our current implementation, we use the most generic option for Ψ, randomly selecting chunks for activation. We show in Section 5 that it suffices to obtain efficient and satisficing planning results, but that domain-specific associations can improve efficiency significantly.

A simple way of improving Ψ is by introducing a weighting factor α, with $\alpha > 1$. With n chunks, m of which are reinforced, a non-reinforced chunk has probability $p = \frac{1}{n+(\alpha-1)m}$ of being

selected as the first chunk, while a reinforced chunk is selected with probability $p_r = \frac{\alpha}{n+(\alpha-1)m}$. The chunks are selected sequentially (without replacement) until the memory capacity is exhausted. We have tested the applicability of the weighting factor by reinforcing chunks that contain unfulfilled goals.

The interplay between forgetting chunks immediately and reactivating chunks based on the weighting factor results in a rehearsal-like process that resembles the rehearsal processes found in different models of human memory (see Section 2).

For now, we have not integrated our PDDLMemory module with an autonomous robot, in order to easily compare the planning abilities with and without the memory module. Therefore, we simulate the dynamic exploration of the environment by randomly activating chunks using a generic Ψ function. Another possibility to select new chunks to be activated is by association, which would require the knowledge base to contain some structure of associations between chunks. We want to explore this possibility in future work and also integrate our module with an autonomous robot making use of actual sensory information.

5 Evaluation

We compared PDDLMemory with planning without memory restrictions, measuring efficiency by planning time and plan quality by plan length. We also investigated the number of iterations needed to fulfill all goals with the simple probabilistic choice of chunks. For all experiments we used the Fast-Forward planner[1] [14] and a memory capacity of four.

For the following experiments, we used problem sizes of one to five randomly selected goals, generating 20 instances per problem size. Figure 4(a) compares the planning times required by the Fast-Forward planner with and without PDDLMemory. In both cases, the planning time increases with the number of goals, but PDDLMemory significantly flattens the gradient, even though it requires more iterations with an increasing number of goals. For five goals, a plan is found about three times faster with PDDLMemory.

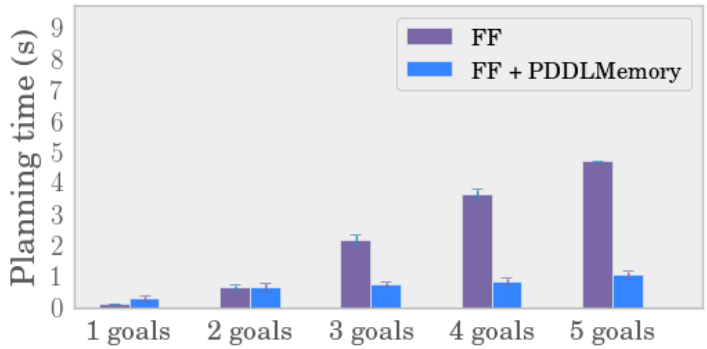
Figure 4(b) compares the lengths of the resulting plans. As PDDLMemory solves the goals in isolation, we had expected an increase in plan length. However, this increase turned out to be very slight. This may be due to our specific domain where the goals are rather independent and the only possible synergies are the movements between places. This phenomenon seems to be characteristic of everyday environments, so that there is practically no loss of plan quality. But this depends on the domain and needs to be confirmed for other domains and problem sizes.

Figure 4(b) also shows the portion of problems that were solved. For this experiment, PDDLMemory was given a maximum of 100 iterations to solve all the goals. In some cases, this was not sufficient and one or more goals remained unsolved, because the relevant memory chunks were not activated in any of the iterations.

We then examined the necessary number of iterations to fulfill all five test goals. Figure 4(c) visualizes the remaining goals against the PDDLMemory iterations. To achieve five goals, about 100 iterations were needed. This number looks quite high, but as the single planning problems are small, the overall run time is only affected slightly (cp. Figure 4(a)).

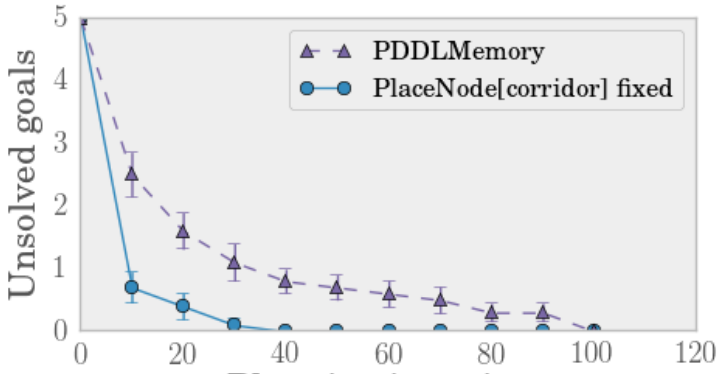
In our domain, the chunk corresponding to the corridor (Fig. 2(a) on page 191) is necessary for many goals, because the robot has to pass through the corridor in order to get utensils from other rooms. To see how much this affects the needed iterations, we ran the same experiment,

[1] http://fai.cs.uni-saarland.de/hoffmann/2002.html



(a) Planning times with the FF planner with and without PDDLMemory.

(b) Plan lengths with the FF planner with and without PDDLMemory. The numbers above the bars show the portion of fully solved problems.



(c) Number of necessary iterations to fulfill five goals with PDDLMemory. The nodes show the average values from 10 runs.

Figure 4: Experimental results. The error bars show the standard error. For PDDLMemory, we compare only plan lenghts of fully solved problems. (In cases in which PDDLMemory fails to solve the complete problem, it still generates partial plans for some of the subproblems. These are usually shorter, so they are excluded from comparison.)

but with the corridor chunk having a probability of 1 to be activated (reducing the memory capacity for other chunks to three). Now the five goals are achieved with 30–40 iterations.

This alteration of always keeping the corridor chunk in memory is very specific to our domain. But it shows that it may be beneficial to make some knowledge universally known. An association structure between chunks could have a similar effect, because the corridor is adjacent to all other rooms and would thus have a high probability of being activated by association.

6 Related Work

The efficiency gain of PDDLMemory is achieved by reducing the state space, an idea that has been used in different methods in AI planning. Hoffmann et al. [15] suggested to use landmarks as a strategy for problem decomposition. A landmark generation graph decomposes a planning task into smaller subgoals and iteratively determining those landmarks that are achievable in the next step, passing them as a disjunctive goal to a base planner. The landmark approach provides good results for some tasks; nevertheless, it often generates very long plans and sometimes it finds no solution for solvable tasks [24].

Sebastia et al. [25] suggest an improved problem decomposition technique, STeLLa, which is also based on landmarks. They obtain plans with similar or even better quality than plans obtained when solving the problem without decomposition. A preprocessing step, however, introduces additional overhead, which for some problems takes more time than solving the original problem.

PDDLMemory is a specific, cognitively inspired case of automatic domain transformation. For instance, Areces et al. [3] introduce a method for splitting large action definitions. In PDDLMemory, actions are currently regarded as monolithic items that are always known. The knowledge representation can be extended to include actions in chunks and possibly use different versions of an action in different chunks to mirror the demands of each place and at the same time accelerate the planning process.

Open-world planning assumes that goals and facts get known during the plan execution. Talamadupula et al. [26] propose open world quantified goals as a means to use standard AI planners in open worlds. In contrast, PDDLMemory supports open-world planning by modularizing goals. When new goals or facts become known, they are added to the inactive memory M_i and may be activated in the next iteration cycles. Thus, new or changed knowledge is smoothly integrated into the iterative solution process. In addition, the execution is supposed to be more reactive by executing subplans as soon as they are known. We only combined the plans into one overall plan in this paper as a basis for comparison.

Related to open-world planning is contingent planning, in which some information has to be actively acquired by the agent as part of the planning process [19]. PDDLMemory follows an optimistic approach, in which the robot will sooner or later sense the necessary information and we have shown that even a simple randomized activation scheme leads to acceptable solutions in our apartment domain.

The Switching Planner [13, 12] combines AI planning with a decision-theoretic planner. In this case, the AI planner is the fast processing step, abstracting from the underlying uncertainties in the world. In critical situations, the decision-theoretic planner takes over to plan with probabilistic states for a limited number of steps into the future. The size of the state space given to the decision-theoretic planner is carefully chosen based on entropy. Here, the problem size is actively controlled for, while in PDDLMemory the problem size is indirectly limited by the number of available chunks and the knowledge modeled into the individual chunks.

7 Discussion and Future Work

The work presented here is a first step to better understand how the notion of a short-term memory can be used in combination with AI techniques such as planning to enable autonomous agents to make useful, timely decisions in everyday situations.

We have shown in a specific household domain that a cognitively inspired memory model significantly accelerates planning without compromising plan quality. The method is aimed at dynamic, open-world environments, where planning efficiency is important and the piecewise fulfillment of small goals is an additional advantage. However, the domain definition requires additional workload to structure the available knowledge into chunks. The generalizability of our results can only be demonstrated when used with different environments, which we plan to do in future work.

In addition, we will use PDDLMemory in a more realistic setup with a household robot that can execute the plans in an uncertain, dynamic world and receive new goals from a user while performing its chores.

With more experience in different domains, a more theoretical basis and analysis of the method should be another future step. Such an analysis could contribute to a better understanding of the types of problems that can profit from this approach, as opposed to more formalized problems that need optimal solutions. A deeper analysis is also necessary to determine adequate sizes and other requirements for chunks. Putting all knowledge into a single chunk would lead to the original large planning problem, whereas disregarding chunks completely and working on the level of single predicates as independent items would increase the number of necessary iterations. Also the definition of chunks can fulfill other roles in a system. For example, a chunked knowledge structure could make the interaction with a user more intuitive.

In our example domain, most of the goals were single-chunk goals. The multi-chunk goal for cleaning the bathroom could also be handled, even with the randomized method for memory management. But we cannot claim that this method scales to intricate domains with many and large multi-chunk goals. However, for our purposes of acting in everyday dynamic worlds, this seems to be acceptable. On the one hand, most everyday activities are indeed rather simple when viewed from an AI planning perspective. For example, in a diary study about routine actions [16], the test person performed only 6 activities (such as preparing food or cleaning the table) in the kitchen every morning over 14 workdays and all activities were done sequentially. On the other hand, errors in everyday activity are usually not disastrous. So in order to achieve human-level intelligence, interference between partial plans can lead to inefficiency, even to errors. But this is acceptable in some domains as long as the errors have no disastrous consequences and the agent can recognize and correct them.

Studies in psychology point to a positive correlation between working memory capacity and general fluid intelligence [7]. However, working memory capacity is very limited for all human beings, thus a generalization to even higher capacities (let's say, ten) could never be tested in humans. For PDDLMemory we chose a short-term memory capacity of four, but our domain was also rather small compared to real everyday environments. For other environments, a higher memory capacity may be more adequate, and it does not have to be in the range of human memory capacity. But we suggest that some limitation of short-term/ working memory capacity has benefits for cognitive agents. For cognitive science, this means that in addition to the available evidence, computational models may help to better understand the role of memory capacity and possibly find indications for reasonable boundaries of it.

Acknowledgements

With the support of the Bavarian Academy of Sciences and Humanities.

References

[1] George A. Alvarez and Patrick Cavanagh. "The capacity of visual short-term memory is set both by visual information load and by number of objects". In: *Psychological science* 15.2 (2004), pp. 106–111.

[2] John R. Anderson. "ACT: A simple theory of complex cognition." In: *American Psychologist* 51.4 (1996), p. 355.

[3] Carlos Eduardo Areces, Facundo Bustos, Martín Dominguez, and Jörg Hoffmann. "Optimizing Planning Domains by Automatic Action Schema Splitting". In: *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*. 2014.

[4] Richard C. Atkinson and Richard M. Shiffrin. "Human memory: A proposed system and its control processes". In: *Psychology of learning and motivation* 2 (1968), pp. 89–195.

[5] Alan D Baddeley and Graham J Hitch. "Working memory". In: *The psychology of learning and motivation* 8 (1974), pp. 47–89.

[6] William G. Chase and Herbert A. Simon. "Perception in chess". In: *Cognitive Psychology* 4.1 (1973), pp. 55–81.

[7] Andrew R. A. Conway, Nelson Cowan, Michael F. Bunting, David J. Therriault, and Scott R.B. Minkoff. "A latent variable analysis of working memory capacity, short-term memory capacity, processing speed, and general fluid intelligence". In: *Intelligence* 30.2 (2002), pp. 163–183.

[8] Nelson Cowan. "An embedded-processes model of working memory". In: *Models of working memory: Mechanisms of active maintenance and executive control* (1999), pp. 62–101.

[9] Nate Derbinsky and John E. Laird. "Effective and Efficient Forgetting of Learned Knowledge in Soar's Working and Procedural Memories". In: *Cognitive Systems Research* 24 (2013), pp. 104–113.

[10] H. Eichenbaum. *Learning & Memory*. W. W. Norton & Company, 2008.

[11] Michael W. Eysenck and Michael C. Anderson. *Memory*. English. 1 edition. Hove England, New York: Psychology Press, Feb. 2009.

[12] Moritz Göbelbecker, Charles Gretton, and Richard Dearden. "A switching planner for combined task and observation planning". In: *In TwentyFifth Conference on Artificial Intelligence*. 2011.

[13] Marc Hanheide et al. "Exploiting Probabilistic Knowledge under Uncertain Sensing for Efficient Robot Behaviour". In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*. Barcelona, Spain, July 2011.

[14] Jörg Hoffmann and Bernhard Nebel. "The FF Planning System: Fast Plan Generation Through Heuristic Search". In: *Journal of Artificial Intelligence Research* 14 (2001), pp. 253–302.

[15] Jörg Hoffmann, Julie Porteous, and Laura Sebastia. "Ordered landmarks in planning". In: *J. Artif. Intell. Res.(JAIR)* 22 (2004), pp. 215–278.

[16] Michael Karg and Alexandra Kirsch. "Low Cost Activity Recognition Using Depth Cameras and Context Dependent Spatial Regions". In: *Workshop on Autonomous Robots and Multirobot Systems (ARMS), Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2014.

[17] Pat Langley, John E. Laird, and Seth Rogers. "Cognitive architectures: Research issues and challenges". In: *Cognitive Systems Research* 10.2 (2009), pp. 141–160.

[18] Pat Langley, Kathleen B. McKusick, John A. Allen, Wayne F. Iba, and Kevin Thompson. "A Design for the ICARUS Architecture". In: *ACM SIGART Bulletin* 2.4 (July 1991), pp. 104–109.

[19] Shlomi Maliah, Ronen Brafman, Erez Karpas, and Guy Shani. "Partially Observable Online Contingent Planning Using Landmark Heuristics". In: *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*. 2014.

[20] Hanspeter A. Mallot and Kai Basten. "Embodied spatial cognition: Biological and artificial systems". In: *Image and Vision Computing*. Cognitive Systems: Perception, Action, Learning 27.11 (Oct. 2009), pp. 1658–1670.

[21] Drew McDermott and the AIPS-98 Planning Competition Committee. *PDDL — the planning domain definition language*. Tech. rep. CVC TR-98-003/DCS TR-1165. Yale University, 1998.

[22] George A Miller. "The magical number seven, plus or minus two: some limits on our capacity for processing information." In: *Psychological review* 63.2 (1956), p. 81.

[23] Klaus Oberauer and Stephan Lewandowsky. "Forgetting in immediate serial recall: Decay, temporal distinctiveness, or interference?" In: *Psychological Review* 115.3 (2008), p. 544.

[24] Silvia Richter and Matthias Westphal. "The LAMA planner: Guiding cost-based anytime planning with landmarks". In: *Journal of Artificial Intelligence Research* 39.1 (2010), pp. 127–177.

[25] Laura Sebastia, Eva Onaindia, and Eliseo Marzal. "Decomposition of planning problems". In: *Ai Communications* 19.1 (2006), pp. 49–81.

[26] Kartik Talamadupula, J. Benton, Subbarao Kambhampati, Paul Schermerhorn, and Matthias Scheutz. "Planning for Human-robot Teaming in Open Worlds". In: *ACM Transactions on Intelligent Systems and Technology* 1.2 (Dec. 2010), 14:1–14:24.