Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference

Rationale-Based Visual Planning Monitors for Cognitive Systems

Zohreh A. Dannenhauer, Michael T. Cox
Wright State University
Dayton, OH
E-mail: alavi.3; michael.cox@wright.edu

Abstract

In this paper, we introduce a new technique for planning in a world under continuous change. We focus on the relationship between vision, interpretation and planning. Our approach is to make vision sensitive to relevant changes in the environment that can affect plans and goals. For this purpose, we applied a rationale-based monitor technique to a hierarchical planner. The planner generates plan monitors that interact with a vision system and react only to those environmental changes that bear on current planning decisions. Thus when the monitors detect these changes, they execute specific plan transformations as needed. We present results with a cognitive architecture using the monitors to focus vision and adapt plans. An experiment in a blocks world demonstrates the effectiveness of our approach.

Introduction

The ability to act and respond to exogenous events in dynamic environments is crucial for robust autonomy. In dynamic environments, external changes may occur that prevent an agent from reaching its goal(s). But the default strategy in most agent systems is to wait until a plan step fails, then repair the plan or start planning anew (Cushing and Kambhampati 2005). Instead, we claim that an intelligent agent should actively watch for what can go wrong and anticipate mistakes before they occur.

Vision has traditionally been a distinct area of research and vision systems act independently of planning and agent behavior. The "goal" of vision is to accept a visual scene as input and to label the objects and perhaps identify the relations between objects as output (see for example (Marr 1982)). Agent goals are irrelevant. Once an agent architecture receives the output of vision, the system can search for objects or relationships that bear on goals and plans. This division of labor is quite inefficient since many of the objects in a visual scene will likely never affect the agent. In contrast, an active approach to vision asserts that the vision system should operate with the goals and plans as guide (c.f., (Findlay and Gilchrist 2003; Fermuller and Aloimonos 1994)). In the research reported here, we propose a novel integration of planning and interpretation that uses goals and plans to bias an active vision component.

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

We introduce a new system for planning in a world under continuous change in an agent with visual perception. Our main contribution is making vision sensitive to relevant changes in the environment during the planning process that affect an agent's plans. We apply a rationale-based monitor technique (Veloso, Pollack, and Cox 1998) to the SHOP Hierarchical Task Network (HTN) planner (Nau et al. 1999). Rationale-based monitors provide a means of focusing visual attention on features of the world likely to affect the plan. We modified SHOP to generate plan monitors to interact with a vision system and react only to those environmental changes that bear on current planning decisions. Thus when the monitors detect any relevant changes, corresponding plan transformations are executed as needed. We discuss when to create the monitors, and when/how to backtrack the planner to modify the plan. We have added our extended SHOP planner to the planning phase of a cognitive architecture named MIDCA, refined the integration with a Baxter robot, and tested it on a simulated blocksworld domain. MIDCA communicates with Baxter to accomplish goals in a dynamic environment using the monitors to focus vision and adapt plans.

We begin the paper by describing the MIDCA cognitive architecture and the SHOP planner embedded within it. The next section describes the vision monitors concept and how it is implemented in the SHOP planner. An empirical evaluation of this technique follows with related research and a conclusion finishing the paper.

The MIDCA Architecture and SHOP

The *meta-cognitive, integrated dual-cycle architecture (MIDCA)* (Cox et al. 2016) consists of "action-perception" cycles at both the cognitive level and the meta-cognitive level. In general, a cycle performs problem-solving to achieve its goals and tries to comprehend the resulting actions and those of other agents. The output side of each cycle consists of intention, planning, and action execution, whereas the input side consists of perception, interpretation, and goal evaluation. MIDCA 1.3 is the latest implementation of the MIDCA architecture. In MIDCA 1.3, we have added an API interface to communicate with the Robot Operating System (ROS) and a humanoid Baxter robot. In problem solving, the Intend component commits to a current goal from those available. The Plan component then generates a sequence of actions. Comprehension starts with perception of the world in the attentional field via the Perceive phase. In this paper, we study the relationship between Plan and Perceive.

The Plan phase in MIDCA uses the SHOP planner. Goals in MIDCA are mapped into initial tasks for SHOP planner. SHOP is an HTN planning algorithm that creates plans by recursively decomposing tasks into smaller subtasks until only the primitive tasks are left which can be accomplished directly (Nau et al. 1999). SHOP uses methods and operators. An operator specifies a way to perform a primitive task, and a method specifies a way to decompose a non-primitive task into a set of subtasks. A planning operator is a triple o = (head(o), pre(o), eff(o)), where pre(o) and eff(o) are preconditions and effects.

An HTN planning problem is a 3-tuple $P = (s, T, D)$. It takes the initial state s and a set of tasks $T = \langle t_1, ..., t_k \rangle$ to be accomplished. Also, it takes a knowledge base D including operators and methods. A plan $\pi = \langle \alpha_1, ..., \alpha_n \rangle$ is a solution for a planning problem to accomplish T.

The Perceive phase in MIDCA generates discrete world states which are represented symbolically as logical predicates on objects in an image. As the Baxters camera reads in images, a visual detection node performs an object detection procedure to locate the objects and sends object data (e.g., color, location) about any known objects to the buffer. The visual detection node is a ROS node that is running concurrently with MIDCA and sends spatial representation of the world to MIDCA. The object detection algorithm detects objects. Then the Perceive phase in MIDCA reads the spatial representation from the buffers and generates a symbolic representation of the world. Perceive stores both spatial and symbolic forms in memory. In the following sections, we explain how these two phases in MIDCA interact.

Rationale-based Vision Monitors

A rationale-based vision monitor (Alavi and Cox 2016; Veloso, Pollack, and Cox 1998) provides a means of focusing visual attention on features of the world relevant to what the agent is trying to do. That is, vision should be informed by planning activity, because a plan represents the intended actions to achieve the agent's goals and therefore contains the objects and states of interest to the agent. Vision should thus monitor states that form the basis (i.e., rationale) of planning choices. When a feature being monitored changes, and the change is detected, we say that the monitor fires. If the planner decides to account for the new change, it will update the plan and alter the planning search. In particular, parts of the plan may be deleted because they have become unnecessary; or new tasks may be added and current ones refined; and prior decisions about how to achieve particular goals may be revisited.

Vision Monitors in SHOP

In this section, we explain how we modify the SHOP planner to adapt search in response to changes during planning. To integrate with rationale-based monitors, two procedures are added to the SHOP planner. First, the monitors are generated when an operator o is added to the current plan π . Second, at each planning cycle, the SHOP planner checks for fired monitors. If a monitor fires, the planner refines the plan.

Algorithm 1 Generate monitors

```
1: procedure $generate\_monitors(o, l, s, mnts)$
2:     for $p$ in $precond(o)$ do
3:         if satisfied($p, s$) then
4:             $mnts \leftarrow (p, l) \cup mnts$
5:         else if $\neg$ satisfied($p, s$) then
6:             $mnts \leftarrow (\neg p, l) \cup mnts$
7:         end if
8:     end for
9: end procedure
```

Algorithm 1 shows the details of monitor generation for the preconditions of an operator. It takes the operator o, the current recursion tree depth l, state s, and a list of monitors $mnts$ as input. Monitors observe features that directly influence π. This includes preconditions of all operators in π. Some of these preconditions will be true when they are added to π; they therefore must be monitored, because, should they become false, π will fail unless additional steps are added. Other preconditions will be initially false; should they become true, then the portions of π that established them may become unnecessary.

Algorithm 2 Check for fired monitors.

```
1: procedure $fired(mnts)$
2:     $fired\_list \leftarrow \langle\rangle$ $i \leftarrow$ perceive the world    $\triangleright$ $i$ is the
   spatial representation of the world
3:     for $(p, l)$ in $mnts$ do
4:         if $i \not\models p$ then
5:             $fired\_list \leftarrow (p, l) \cup fired\_list$
6:         end if
7:     end for
8: end procedure
```

Algorithm 2 checks to see if the preconditions of all operators in the plan so far are still satisfied in the new perceived state. It gets the spatial representation i of the world and check to see if the precondition can be extracted form i.

Plan refinement is done by backtracking to the recursion tree depth that an action fails. When monitors are generated, the current recursion depth is recorded and backtracking uses this information.

Plan Refinement The core of our approach is how to refine the plan under construction with backtracking and altering the task-decomposition. When an operator fails (the preconditions of the operator are not met in the new state), the modified SHOP planner backtracks to the depth that the failed operator is added to the plan. The refining procedure starts with traversing the parent links until it finds the closest valid parent of the failed task. A valid parent is a non-primitive task which has been decomposed by a method for which all preconditions of that method are true in the new state.

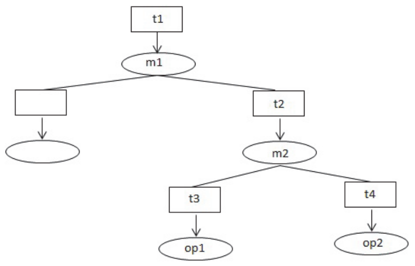

Figure 1: An example of task decomposing

Figure 1 shows an example of decomposing task t_1. Consider in the middle of planning task t_3 is failed (the preconditions of operator p_1 are not met). The SHOP planner will backtrack to t_3 to refine the plan. To check if t_2 is a valid task, the planner checks if the preconditions of the method m_1 are true, because t_1 is decomposed to t_2 using method m_1. If there is a precondition of method m_1 that is not satisfied in the currently observed state of the world, then this means t_2 is also a failed task. It continues the process until it finds a task that is valid in the current state or it reaches the goal task. When the algorithm finds a valid task it tries to decompose it in another way and continues building the rest of the plan.

A Vision Application

We have added our extended SHOP planner to the planning phase of MIDCA_1.3 and tested our system with a Baxter humanoid robot in a blocksworld domain to examine our claim on focusing vision and relationship between planner and perception.

Monitors are created during plan generation as actions are added to the plan. Monitors store the preconditions of the actions. These monitors are mapped to a corresponding component in Perceive that is only concerned with state changes related to that specific precondition. We used expert authored perceptual functions for perceiving each predicate (on, clear, etc.) and mapped these to monitors in the planner. The planner checks to see if the observed state is the same as the expected state. If any change happens, the planner refines the plan based on the new state.

Empirical Evaluation

In this section, we describe our experiment with MIDCA on a modified blocksworld domain. We use a standard simulator to simulate the world state and actions.

Our work focuses on the relationship between the Perceive and Planning phases. The Perceive phase assists the Plan phase by monitoring the relevant features of the world during planning time. Relevant features are those that relate to the current plan. In the evaluation below, we show how vision assists planning. We leave the evaluation of planning assisting vision to future work.

Blocksworld Domain

To evaluate the performance of our approach to the relationship between vision and planning, we ran the system in a modified blocksworld domain. The goal of this experiment is to examine the benefit from using vision monitors to improve planning in a dynamic environment. This version of blocksworld domain (Winograd 1971) includes rectangular blocks. The initial tasks for problems in this domain are to build houses consisting of towers of blocks. We added the possibility that blocks could catch fire and before any block was picked up, the fire should first be extinguished. In order for an extinguisher to be used, it must first be taken out of the box. The box itself is represented as a block. If the box is not clear, the planner generates a plan to make the box clear. Furthermore, there were additional actions available to MIDCA allowing it to deal with these refinements. The three new types of actions are as follows:

put-out-fire(A, ext**)** If A is on fire, extinguish A

get-extinguisher(ext**,** B**)** if B is clear, take out the extinguisher ext from B

make-box-clear(B**)** if B is not clear, unstack all blocks on top of B

Experimental Results

In this experiment, we changed the world state in the middle of planning to make the current plan not valid. Our hypothesis is that the planner will refine the plan to successfully accomplish the given tasks.

In each planning problem we set the initial state to be one with a block A on fire, a separate tower with C as its bottommost block, and a fire extinguisher, ext, inside C. The goal is on(A, B). In order to pickup A, the fire needs to be extinguished first. For example, if the height of tower is 3, the planner has to unstack and putdown 2 blocks in order to obtain the fire extinguisher from block C and use it on block A. If the fire goes out during the planning process, the monitor watching the precondition onfire(A) fires. Then the planner cuts parts of the plan related to putout fire and simply generate the plan pickup(A),stack(A, B).

Here, the purpose of monitoring is to observe such a change as the fire going out, and suggest a cut in the plan. By varying the height of the tower, we can vary the complexity and length of the solution. In this experiment, we varied the height of tower, n, from 10 to 100 in increments of 20. During planning the monitor which observes the state of onfire(A) fires and suggests a plan refinement. We vary the time at which this monitor fires during the planning process, namely after 10, 70, 100, 200 planning steps.

Figure 2 shows the results of the experiment and plots the planning steps as a function of n. As can be seen, when the environment does not change, the number of planning steps increases with n. However, with the rationale-based monitors, the planner can react to the state changes and find a solution faster. As would be expected, when the changes occur later, the savings benefit of the planner is reduced, because it has already performed significant planning. When the delay is infinite, it has to unstack all the blocks to get the extinguisher. When the delay equals 10, the fire goes out in the very beginning and the number of planning steps is 20 for all the tower height.

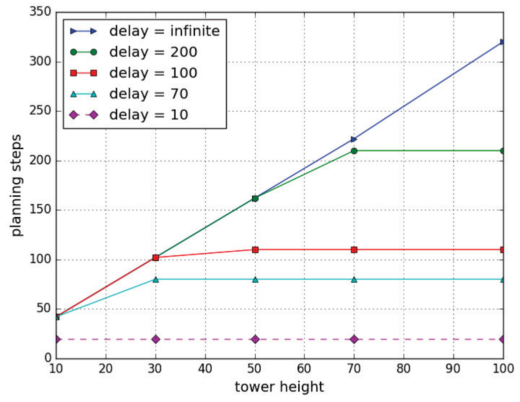

Figure 2: Planning performance using rationale-based monitors in the SHOP planner. The curves refer to different delays of the state change during the planning process.

Related Work

Similar work has been previously performed by (Veloso, Pollack, and Cox 1998). They implemented rationale-based monitors in the state space planner Prodigy, whereas our approach differs in using these monitors in the SHOP HTN planner. Our next step is to examine the benefit of using these monitors during the act (execution) phase. Our method allows the agent to respond to unexpected changes during execution (c.f., (Pettersson 2005)). For example in (Ayan et al. 2007), the authors introduce an HTN-based planning system which revises the plan if any action fails due to a state change. Our approach lets the agent know about action failure earlier, so it has a chance to revise the plan sooner before reaching the failed action. In real world examples (e.g., military logistics scheduling), planning duration can be extensive, so significant changes often occur in the interval.

Conclusion

The integration of planning and interpretation in a cognitive architecture is not a simple one way interaction. Here we have argued that vision should serve the needs of the planner. The planner generates visual monitors for the vision system based on the rationale for plan decisions (e.g., preconditions), and the vision system detects when these conditions are violated. However, it can equally be argued that the planning component should serve the needs of vision and interpretation. Given a particular scene or situation, MIDCA's interpretation component recognizes new problems in terms of expectation failures or discrepancies. The interpretation system will then attempt to explain the discrepancy and use the explanation to generate a goal to remove the problem. The goal is passed to the problem-solving module of MIDCA, and the planner will generate a plan to achieve it. This technique typifies the goal-driven autonomy approach to goal reasoning (e.g., (Aha et al. 2010; Klenk, Molineaux, and Aha 2013) Our results support the idea that vision has an important role in supporting the intentions and actions of the agent. However, much future work remains to be performed.

Acknowledgments

This research is funded by ONR under grants N00014-15-C-0077 and N00014-15-1-2080. We also thank the anonymous reviewers for their comments and suggestions.

References

Aha, D.; Klenk, M.; Munoz-Avila, H.; Ram, A.; and Shapiro, D. 2010. Goal-driven autonomy: Notes from the aaai workshop.

Alavi, Z., and Cox, M. T. 2016. Rationale-based visual planning monitors. In *Working Notes of the 4th Workshop on Goal Reasoning. New York, IJCAI-16.*

Ayan, N. F.; Kuter, U.; Yaman, F.; and Goldman, R. P. 2007. Hotride: Hierarchical ordered task replanning in dynamic environments. In *Planning and Plan Execution for Real-World Systems–Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop. Providence, RI*, volume 38.

Cox, M. T.; Alavi, Z.; Dannenhauer, D.; Eyorokon, V.; and Munoz-Avila, H. 2016. Midca: A metacognitive, integrated dual-cycle architecture for self-regulated autonomy. In *AAAI*.

Cushing, W., and Kambhampati, S. 2005. Replanning: A new perspective. *Proceedings of the International Conference on Automated Planning and Scheduling. Monterey, USA* 13–16.

Fermuller, C., and Aloimonos, Y. 1994. Vision and action. *IVC*.

Findlay, J. M., and Gilchrist, I. D. 2003. Active vision: The psychology of looking and seeing.

Klenk, M.; Molineaux, M.; and Aha, D. W. 2013. Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence* 29(2):187–206.

Marr, D. 1982. A computational investigation into the human representation and processing of visual information. *Vision* 125–126.

Nau, D.; Cao, Y.; Lotem, A.; and Muñoz-Avila, H. 1999. SHOP: Simple hierarchical ordered planner. In *Proceedings of the 16th IJCAI-Vol. 2*, 968–973. Morgan Kaufmann.

Pettersson, O. 2005. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems* 53(2):73–88.

Veloso, M. M.; Pollack, M. E.; and Cox, M. T. 1998. Rationale-based monitoring for planning in dynamic environments. In *AIPS*, 171–180.

Winograd, T. 1971. Procedures as a representation for data in a computer program for understanding natural language.