Abstract

Oscar Sapena, Eva Onaindía

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Spain
{osapena,onaindia}@dsic.upv.es

Introduction

Off-line planning generates a complete plan before any action starts its execution [3]. This forces to make some assumptions that are not possible in real environments like, for example, that actions are uninterruptable, that their effects are deterministic, that the planner has complete knowledge of the world or that the world only changes through the execution of actions. On the other hand, on-line planning allows to start execution while the planner continues working in order to improve the overall planning and execution time. Nowadays there are only some few approaches for planning in dynamic environments and/or with incomplete information [2]:

- *Conditional planning*: this approach tries to consider the possible contingencies that can occur in the world.
- *Parallel planning and execution*: this approach separates the planning process from the execution.
- *Interleaving planning and execution*: this approach allows quick and effective responses to changes in the environment.

SimPlanner is an integrated tool for planning and execution, and it is based on this latter approach. This tool is thought to be used in real environments such as the intelligent control of robots. However, it has initially been implemented as a simulator in order to check its behavior without having to integrate it in different several domains. SimPlanner is made up of three components: an on-line planner, a monitoring module and a replanner.

The on-line planner

The on-line planner is responsible for generating, in an incremental way, a plan to achieve the goals. As soon as the planner calculates the first action, the plan can start its execution. From this moment on, the planning and execution processes keep on working in parallel while no unexpected event is detected; otherwise, the execution must wait for the planner to make the necessary modifications in the plan.

The planner is based on a depth-first search, with no provision for backup. The planning decisions (inferred actions) are consequently irrevocable. The planning algorithm uses heuristic functions to compute an approximate plan (AP)

for each goal independently. Then, a conflict-checking mechanism detects conflicts among actions in the approximate plans and selects the action from the AP that minimizes the number of conflicts. The selected action is inserted at the end of the plan, and the current state is updated through its execution. This algorithm is iteratively executed until all top-level goals are achieved [4].

The monitoring module

Monitoring is the process of observing the world and trying to find discrepancies between the physical reality and the beliefs of the planner [1]. Basically, there exists two types of plan execution monitoring [2]: *action monitoring* checks the validity of the action preconditions before it starts its execution and also that its effects have taken place as expected. The *environment monitoring* tries to capture information from the external world that conditions the remaining planning process. Monitoring is, therefore, domain-dependent. Since SimPlanner is being used at the moment as a simulator, this information is input to the system by the user. The user is who decides what information the robot receives and which unexpected events that occur in the world are communicated.

The replanner

When an unexpected event is detected, the calculated plan is checked in order to assure it is still valid [1]. If this is the case, the execution simply continues. Otherwise the replanning module is invoked. The replanner tries to reuse as much of the calculated plan as possible without losing the quality of the final plan. It uses a heuristic function to find out which is the best reachable state through the actions in the original plan. If there are many reusable actions, the planning process will save a lot of computation time. In the worst case, a new plan will be computed from scratch. The replanner overhead is very small so it is worth trying to reuse the plan rather than planning from scratch [5].

References

1. G. De Giacomo and R. Reiter, 'Execution monitoring of high-level robot programs', *Principles of Knowledge Representation and Reasoning*, 453–465, (1998).
2. K.Z. Haigh and M. Veloso, 'Interleaving planning and robot execution for asynchronous user requests', *Papers from the AAAI Spring Symposium*, 35–44, (1996).
3. M.E. Pollack and J.F. Horty, 'There's more to life than making plans: Plan management in dynamic, multi-agent environments', *AI Magazine*, **20(4)**, 71–84, (1999).
4. O Sapena and E. Onaindia, 'Domain-independent on-line planning for STRIPS domains', *Proceedings of Iberamia-02, LNAI*, **2527**, 825–834, (2002).
5. O Sapena and E. Onaindia, 'Execution, monitoring and replanning in dynamic environments', *Workshop on On-Line Planning and Scheduling, AIPS-02*, (2002).