An Integrated Planning and Learning Framework for Autonomous Mobile Robots

C. Ugur Usug, Dogan Altan, Sertac Karapinar, Mustafa Ersen and Sanem Sariel-Talay
Artificial Intelligence and Robotics Laboratory
Istanbul Technical University, TURKEY

http://air.cs.itu.edu.tr

The Scope

Handling Temporary Execution Failures

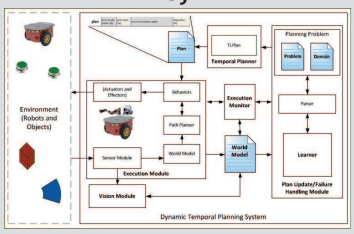
We propose a dynamic temporal planning framework to handle temporary action execution failures at runtime. Some temporary failures may not be resolved by replanning. This research focuses on developing methods for enabling replanning to resolve temporary failures.

Introduction

When the real outcomes of actions are not completely represented in the planning domain, a planner may not be able to construct a valid plan even if there exists one. This research focuses on generic domain update and reasoning methods to construct alternative plans against real-time execution failures that are detected either during runtime or earlier by a plan simulation process. Based on the updated domain representations, a new executable plan is constructed even when the outcomes of existing operators are not completely known in advance or valid plans are not possible with the existing representation of the domain.

Integrated Planning and Learning Framework

The System



TLPlan Temporal Planner

TLPlan, is a forward chaining temporal planner originally proposed by Bacchus and Ady (2001). A search node in TLPlan includes the world state and the action applied along with the world clock that defines the start time of that action. Applying an action at a state means that this action is scheduled for the world clock of that state. The search proceeds by applying new actions or advancing the world clock toward finding a complete temporal plan. TLPlan is used as the temporal planner in this system since it can construct plans for durative and concurrent multirobot actions.

Reasoning on Failures

Methods 1–3 ensure that alternative plans are constructed to resolve temporary failures. These plans are constructed by considering actions that may change the properties of the cause of failure. Integrating a reasoner is a crucial part of the ongoing work to provide a more systematic way to handle failures. Reasoning is useful to eliminate irrelevant actions that are selected by the replanning process. Furthermore, the knowledge-base is incrementally expanded.

Method - 1

```
(def-adl-operator (move-to-loc ?loc)
   (pre
      ...
      (not (move-to-loc_locked ?loc))
   )
   ...
)

(def-adl-operator (pseudo1 ?obj ?loc)
   (pre
      ...
      (and
            (= ?obj obstacle)
            (move-to-loc_locked ?loc)
            (or
                  (not (xcoord ?obj obj_x))
                  (not (ycoord ?obj obj_y))
            )
      )
   )
   (del (move-to-loc_locked ?loc))
)
```

The operator related to the failed action

A pseudo operator is added to the domain

Properties of the cause of the failure are represented in the preconditions of the operator.

Method - 2

```
(def-adl-operator (move-to-loc ?loc)
   (pre
      ...
      (not (move-to-loc_locked ?loc))
   )
   ...
)

(def-adl-operator (push-2 ?obj ?loc)
   (pre
      ...
      (and
            (= ?obj obstacle)
            (move-to-loc_locked ?loc)
            (or
                  (not (xcoord ?obj obj_x))
                  (not (ycoord ?obj obj_y))
            )
      )
   )
   ...
   (del (move-to-loc_locked ?loc))
)
```

Copies of the operators that may change the state of the cause of failure are added to the domain.

Additional preconditions are set for these operators.

Method - 3

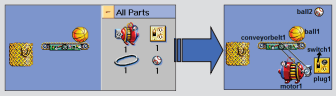
```
(def-adl-operator (move-to-loc ?loc)
   (pre
      ...
      (not (move-to-loc_locked ?loc))
   )
   ...
)

(def-adl-operator (move-to-loc-2 ?loc ?obj)
   (pre
      ...
      (and
            (= ?obj obstacle)
            (move-to-loc_locked ?loc)
            (or
                  (not (xcoord ?obj obj_x))
                  (not (ycoord ?obj obj_y))
            )
      )
   )
   ...
)
```

A copy of the operator related to the failed action is added to the domain.

Additional preconditions are set for this operator.

Learning Action Schema

When the planning agent lacks complete information about the planning domain, the agent needs to learn these representations from observations. In an ongoing work, we develop a method for learning action schema through observations of a given sequence of actions. We have selected the Incredible Machine as a domain for analyzing these interactions because this domain allows the use of various objects and relations to achieve a given goal.



Experimental Results

Planning Domain Experiments

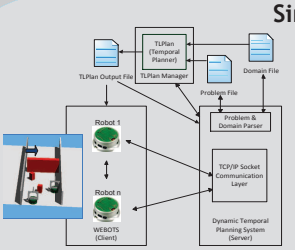
Domain Operators

(move-to-loc ?robot ?location)
(move-to-obj ?robot ? location)
(pick ?robot ?smallObject)
(drop ?robot ?smallObject)
(push ?robot ?largeObject)

Problem Instance	Method	Best-first Search				A*			
		#GN	#EN	Plan Time (s)	Makespan (s)	#GN	#EN	Plan Time (s)	Makespan (s)
1 robot 1 box	M1	46	23	0,038	46,97	103	54	0,088	46,97
	M2			N/A		123	68	0,116	46,97
	M3	44	22	0,04	46,97	79	45	0,078	46,97
1 robot 2 box	M1	105	45	0,11	96,77	1148	550	0,78	81,01
	M2			N/A		1050	558	0,8	81,01
	M3	100	44	0,106	96,77	782	421	0,868	81,01
1 robot 3 box	M1	177	70	0,194	143,38	21586	9720	59,25	118,4
	M2			N/A		19569	9779	50,55	118,4
	M3	175	69	0,194	143,38	14794	7425	25,3	118,4
2 robot 2 box	M1	128	48	0,136	76,08	35193	12202	170,1	45,13
	M2			N/A		38128	14376	195,13	45,13
	M3	133	50	0,144	76,08	26793	10328	75,58	45,13
2 robot 3 box	M1	319	101	0,323	151,19	N/A			
	M2			N/A		N/A			
	M3	296	97	0,31	151,19	N/A			
2 robot 4 box	M1	609	172	0,586	240,43	N/A			
	M2			N/A		N/A			
	M3	618	174	0,634	240,43	N/A			

GN: Number of nodes generated
EN: Number of nodes expanded
2,53 GHz / 4GB RAM
Max Time: 500s
Memory limitation: 20000 nodes

Simulation Experiments

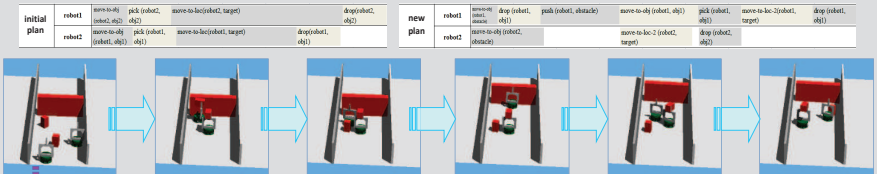


Settings

- Webots Simulator with ODE support
- Simulated Khepera II Robots with grippers
- TCP/IP Communication with the Simulator and the Planner

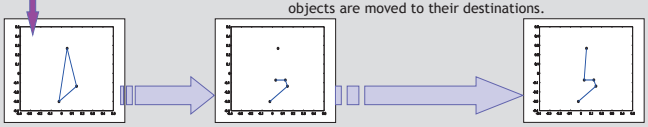
Domain Operators

(move-to-loc ?robot ?location)
(move-to-obj ?robot ? location)
(pick ?robot ?smallObject)
(drop ?robot ?smallObject)
(push ?robot ?largeObject)



Initial plan is executed.

The required domain updates are made due to the failure and a new plan is generated. The new plan involves pushing the obstacle to an appropriate location before the red objects are moved to their destinations.



Topological map is updated.

The overall plan execution of moving red objects to their destinations: (top) plan execution in Webots (bottom) map updates for the first robot.

Conclusion

The proposed approach can efficiently handle temporary failures. Appropriate domain updates are made to replan and generate alternative executable plans even when advanced reasoning tools are not available. An example failure resolution scenario in the Webots simulator is given to validate the proposed approach.

Future Work

Our ongoing work includes an extended set of experiments on several scenarios with Pioneer 3DX robots. The future work includes extending the approach to handle permanent execution failures.