A Possibilistic Planner that Deals with Non-Determinism and Contingency

Emmanuel Guere, Rachid Alami

A Possibilistic Planner that deals with non-determinism and contingency

Emmanuel Guéré and **Rachid Alami**
LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex 4 - France
e-mail : {Emmanuel.Guere, Rachid.Alami}@laas.fr

Abstract

This paper proposes a new planning approach that authorizes an autonomous robot to reason about the inaccuracy of the world description and of its possible evolutions. We represent the uncertainty with the possibility theory; this allows us to distinguish between two types of non-determinism: a non-determinism from insufficient modeling and a non-determinism from uncertainty. Besides, we introduce perception actions as well as a model of the environment dynamics through "contingent events". Finally, we present an implemented experimental planner, based on Graphplan search paradigm. This planner is able to produce plans that are robust with respect to contingent events, and whose goal-achieving ability is evaluated a priori. The obtained plans can be conformant or conditional depending on the context and the user requirements.

1 Introduction

An autonomous robot system operating in an environment in which there is uncertainty and changes, needs to combine reasoning and reaction. To correctly plan in an uncertain and dynamic environment, the planner needs an accurate description of the world, of the actions which could be planned and a goal success definition. There is also a need to model non-determinism as well as contingent events.

One can classify actions into four types: deterministic actions (there is only one possible outcome), conditional actions (outcomes are context dependent), non-deterministic actions (several possible outcomes) and perception actions (one outcome out of n). In addition a perception action improves knowledge.

Another key feature is the ability of the planning algorithm to concentrate on the exploration of the most "probable" courses of action and to evaluate the robustness and the goal achieving ability of a given plan.

This paper presents a planner which partially fulfills the requirements discussed above. To model the environment uncertainty we use the possibility theory [Dubois and Prade, 1988] which allows to represent complete ignorance as well as qualitative inaccuracy. A goal is described as a conjunction of possibilistic facts which can qualify the goal achievement necessity. It is important to note that with uncertain information, actions with conditional effects can be used as a class of actions with non-deterministic outcomes. Consequently non-determinism can result from uncertainty (conditional actions) as well as from insufficient modeling (non-deterministic actions). Our planner uses explicitly these two kinds of sources of non-determinism in planning as well as perception actions which allow to build conditional plans.

Our planner can anticipate and oppose contingent events by avoiding situations where they can occur or by preparing an adapted reaction. However, this feature is limited to situations which are the result of previous robot action.

In the next sections we describe our representation of the world, of the robot actions and of contingent events. Then we present an algorithm inspired from Graphplan search[Blum and Furst, 1997] as well as some illustrative outputs produced by our planner. Finally, we give an overview and a short conclusion.

2 An uncertain environment

It is certainly useful to distinguish between several types of uncertainties: the complete ignorance of a fact (e.g. the robot does not know where the red test tube is), the qualitative inaccuracy of a fact (e.g. the robot only knows that the red test tube is probably on $Table_1$) and the quantitative inaccuracy of a fact (e.g. the robot knows that the probability that the red test is on $Table_1$ is 0.9). Some planners like CGP [Smith and Weld, 1998] or Cassandra [Pryor and Collins, 1996] use a set of distinct possible worlds which corresponds to our complete ignorance idea. The probabilistic approach, used especially in BURIDAN [Kushmeric *et al.*, 1995], is particularly interesting for describing the uncertainty associated to state transitions. However, probabilities do not allow to represent complete ignorance; besides, there are numerous situations where it is not possible to give to the robot planner probabilities based on statistical measures, but only qualitative information provided by the

operator or deduced from previous missions. This is the reason why we use the possibility theory.

2.1 Possibility background

The possibility theory [Zadeh, 1978; Dubois and Prade, 1988] offers an uncertainty modeling framework where two values are associated to every fact A:

- $\Pi(A)$: the "possibility" for the fact A to be true,
- $N(A)$: the "necessity" for the fact A to be true.

The duality between necessity and possibility is expressed by the relation $N(A) = 1 - \Pi(\neg A)$. For example if we do not know if the door is open or not, we can write: $\Pi(Open(Door)) = 1$ and $N(Open(Door)) = 0$ (the door may be open but not necessarily). If the red test tube usually is on $Table_1$, we can write: $\Pi(On(RedTestTube, Table_1)) = 1$ and $N(On(RedTestTube, Table_1)) = 0.9$. Note that this is a qualitative measure; only the order between the different possibility and necessity values is relevant.

2.2 The world representation

We define a state as a conjunction of possibilistic facts. The Closed-World Assumption allows, when a fact A is false, to delete it from the state description. Consequently we extend this assumption, in our context, and we only represent in the state the facts that are known in necessity terms; in other words we only insert facts which have a positive necessity. Table 1 shows an example of an initial state. Note that the possibility is missing; indeed, with the duality between possibility and necessity we could replace $\Pi(A)$ by $N(\neg A)$. So if $\Pi(A) = 1$ then $N(\neg A) = 0$ and it will not be represented in the state description. In this example, we see that the light on top of $Table_1$ often works, whereas the light on top $Table_2$ almost usually works.

Fact	Necessity
On($RedTestTube, Table_1$)	1
On($GreenTestTube, Table_1$)	1
At($Robot, Table_1$)	1
Light($Table_1$)	0.7
Light($Table_2$)	0.9
...	...

Table 1: Example of an initial state description

A goal for our planner is represented by a conjunction of facts with a strictly positive necessity (to avoid disjunctions).

3 Modeling robot actions

We model the robot actions by possibilistic rules that change the state description. These rules are based on an extension of STRIPS [Fikes and Nilsson, 1971] Preconditions, Add-lists and Delete-lists. The computation of the necessities associated to action effects is based on the following propagation rule (fig. 1):

$$N(B_i) \geq max(N^b(B_i), min(N(A_1), N(A_2), ..., N(A_n), N(B_i \mid A_1 \wedge ... \wedge A_n)))$$

where B_i is the i^{th} effect, $N(B_i \mid A_1 \wedge ... \wedge A_n)))$ is the necessity of having B_i given the precondition $A_1 \wedge ... \wedge A_n$, $N^b(B_i)$ is the necessity of the fact B_i before applying the action and $N(B_i)$ is its value after applying the action.

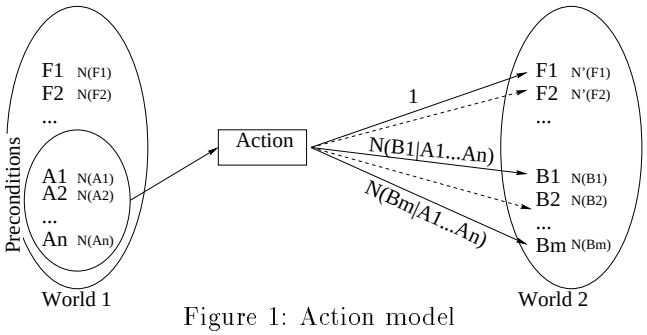

Figure 1: Action model

3.1 Deterministic actions

A deterministic action changes a state into another in foreseeable way. For example the *TakeWithLight* action may be represented by:

TakeWithLight (X:Test Tube, Y:Table)		
Pre:	At Robot Y,	On(X,Y),
	Clear(X),	HandEmpty
	Light(Y)	
Eff:	(¬On(X,Y),1),	(Holding(X),1),
	(¬HandEmpty,1),	(¬Clear(X),1)

If we apply this action in the initial state described in Table 1, we obtain *Holding(RedTestTube)* with necessity $N \geq 0.7$. Note that because there is uncertainty, the action may be executed in situations where its preconditions are not satisfied. We assume that in a such case, it will have no effect. However, if a more sophisticated model is necessary, one can use conditional actions.

3.2 Non-deterministic actions

Non-deterministic actions may lead to several possible outcomes. An outcome is represented by a conjunction of facts (with their necessities) and a possibility-necessity measure of occurrence of that outcome (fig. 2); all possible outcomes must appear (to have N(all outcomes)=1)

To keep consistency, the different worlds w_i that result from the application of a non-deterministic action must verify:

$$\exists i | \Pi(w_i) = 1 \wedge \forall j \neq i, N(w_j) = 0$$

For example, when the *TakeWithoutLight* action is applied, we are sure to hold a test tube, but we do not know if it is Red or Green:

TakeWithoutLight (X:Table)		
Pre:	At Robot X,	HandEmpty
Eff:	(w_1,$\Pi = 1$,N=0):	(Holding(RedTestTube),0.7),
		(¬HandEmpty,0.7)...
	(w_2,$\Pi = 1$,N=0):	(Holding(GreenTestTube),0.7),
		(¬HandEmpty,0.7)...

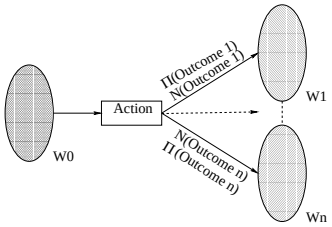

Figure 2: Non-deterministic action model

3.3 Conditional actions

For conditional actions, the effects are context dependent (fig. 3). The conditions must be exhaustive and mutually exclusive. Note that we allow action definitions which combine different types of effects (deterministic, conditional,...). The *Take* action combines conditional and non-deterministic effects:

Take (X:Test Tube, Y:Table)		
Pre:	At Robot Y,	On(X,Y),
	Clear(X),	HandEmpty
Eff:		
When Light(Y):		
	$(\neg On(X,Y),1)$,	(Holding(X),1),
	(¬HandEmpty,1),	(¬Clear(X),1)
When $\neg Light(Y)$		
	$(w_1, \Pi = 1, N=0)$:	(Holding(RedTestTube),0.7), (¬HandEmpty,0.7)...
	$(w_2, \Pi = 1, N=0)$:	(Holding(GreenTestTube),0.7), (¬HandEmpty,0.7)...

In classical logic an action with conditional effects is deterministic. Indeed when we plan this action we know the current state and we can decide which conditional effects will apply. With such a model, one can always change an action which has n conditional effects, each with m conjuncts, in 2^{nm} actions[1]

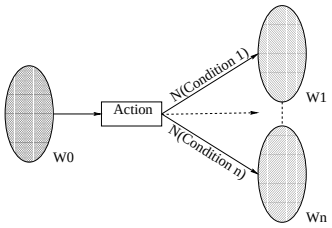

Figure 3: Conditional effects action model

However, in our state model the different facts are possibilistic, so if the fact $Light(Table_1)$ is not completely certain $\neg Light(Table_1)$ is possible (if $\Pi(Light(Table_1)) = 1 \wedge N(Light(Table_1)) = 0.7$ then $\Pi(\neg Light(Table_1)) = 0.3 \wedge N(\neg Light(Table_1)) = 0$). We are in one of two possible worlds; the *Take* action, because of its conditional effects, is applicable in both worlds with different effects. This is what we call non-determinism due to uncertainty.

[1]For more details about conditional effects associated to Graphplan search see [Koehler *et al.*, 1997; Anderson *et al.*, 1998].

3.4 Perception actions

Perception actions allow to improve the robot knowledge about a given set of facts. It will allow the robot to know on which branch of the course of actions it is (fig. 4). This will entail a higher necessity associated to the facts that have been observed. In the current implementation, we use a simple hypothesis: the perception is assumed perfect, and the necessity associated to the outcome is 1. One possible improvement can be the use of conditional effects (C-BURIDAN [Draper *et al.*, 1994] models that an observation can fail).

For example the perception action which determines if the robot has taken the red or the green test tube, can be represented by:

WhichOne?(X:Table)		
Pre:	At Robot X, Light(X)	¬HandEmpty
Eff:	$(w_1, \Pi = 1, N=0)$:	(Holding(RedTestTube),1)
	$(w_2, \Pi = 1, N=0)$:	(Holding(GreenTestTube),1)

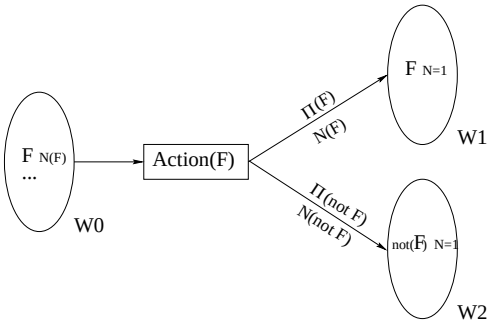

Figure 4: A perception action example

4 An algorithm based on Graphplan

Graphplan is a fast planner developed by [Blum and Furst, 1997] which plans with STRIPS operators and uses a constraint propagation method. It performs in two steps: first, it builds a constraint graph expansion; and then it searches for the plan with a constraint resolution extraction.

4.1 Possibilistic treatment

To compute the necessity of a fact in a given state we apply the propagation rule defined in section 3. The use and definition of mutex is the same as Graphplan except for "interference": two actions, A_1 and A_2 are mutually exclusive if the A_1 effects contain the fact m, A_2 preconditions contain $\neg m$ and $min(\Pi(m), \Pi(\neg m)) > 1 - N(Goal)$. One advantage of Graphplan is to keep on a level only one specimen of a fact. Consequently if we want to keep this advantage, we must use at each level the most "pessimistic" path (lowest necessity) as for proposition mutex: if the fact A has the necessity 0.9 with a first path and 0.8 with a second one, we associate 0.8 to A (see for example $AtRobotTable_1$ in figure 5). The real fact necessity will be reevaluated during the Graphplan backward phase. With a Graphplan algorithm we cannot optimize the plan necessity (Graphplan

only optimizes the number of levels), but the necessity that we associate to the arcs (induced by the action models) can be used as a heuristic during backtrack.

4.2 Non-deterministic treatment

During the graph expansion, the planner applies actions with n non-deterministic effects as n deterministic actions (but we associate a label with necessity and possibility to each branch as shown in figure 5). The solution

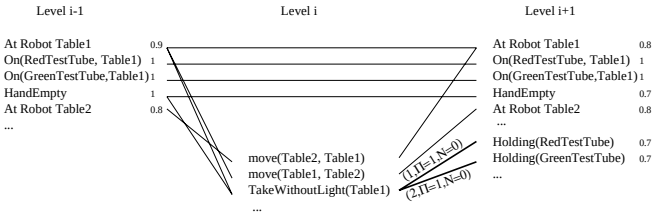

Figure 5: A graph expansion with non-deterministic action (to simplify delete-lists are not drawn)

extraction is more complex; if the plan contains a non-deterministic branch then either the branch necessity is better than the goal necessity (and the plan is valid), or the branch necessity is lower. In this case, the planner must verify that the plan is also valid for the other branches.

For example on figure 6, the planner applies a non-deterministic action *Take* which "creates" two possible worlds w_{11} and w_{21}. At this point, the robot will not be able to distinguish in which world it is. The next actions should be applicable both in w_{11} and w_{21} (e.g. $Move(Table_1, Table_2)$) until a perception action allows to distinguish between the two "branches".

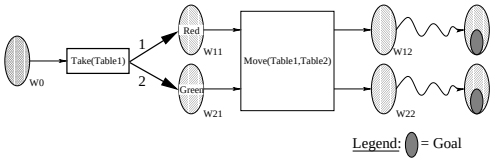

Figure 6: A plan with non-deterministic action

4.3 Perception treatment

The main difference between non-deterministic and perception action is that the robot will know after perception on which "branch" it is. The planner can then build a conditional plan. For example, in figure 7, the planner inserts a perception action $WichOne?(Table_2)$ to distinguish if the robot is holding a red or a green test tube. Note that $WichOne?(Table_2)$ must be applicable in w_{12} and w_{22}. After the $WichOne?(Table_2)$ action, the two plan "branches" are independent and involve different actions. This branches will be explored sequentially. This exploration is not necessary exhaustive. The planner will stop when it finds a plan that satisfies the goal necessity.

This corresponds to two plan synthesis starting from w_{13} and w_{23}. Note that, the planner can re-use, in the second branch exploration, the graph expansion built during the first one. However, it will have to extend it.

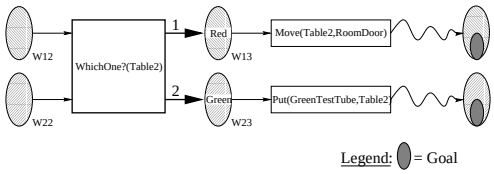

Figure 7: A plan with a perception action

4.4 A first step towards dealing with contingent events

A key requirement for robot planning is to be able to reason about contingencies.We have introduced in our planner a capability that allows to deal with a subclass of external events.

Indeed, we distinguish between "actions" and "events"; while actions are operators over which the robot has full control, events are triggered whenever there is a situation that satisfies their preconditions. This model of the environment dynamics is based on extension of the NODEP representation of contingent events [Perret and Alami, 1995]. Similarly to action types, we have defined events with possibilistic, non-deterministic or conditional effects.

Obviously, an elaborated reasoning on external events imposes temporal reasoning capabilities which are not included in our planner. However, we have chosen to exploit the number of plan steps as an estimation of durations.

We distinguish two types of events:

- "immediate events" i.e events which apply immediately after the situation which verifies their preconditions (one level in Graphplan search),
- events with delay: an event with a delay of n time units will occur if its triggering situation lasts during n levels.

For example, our goal is to cool the red test tube. At the end of the plan, the red test tube must be at $Table_3$ in the cold-room. But if the door stays open, will the cold-room continue to cool? After a moment the test tube which is in the cold-room will be damaged. This is modeled by an undesirable fact called $Fail$,

Such an event can be modeled by:

HighTemperature
Pre: Opened(Door)
Eff: After(10,(Fail,Π=1,N=0.7))

*After(10,(Fail,*Π=*1,N=0.7))* means that after 10 time units the effect Fail (with necessity N=0.7) will occur.

A "reasonable" robot should avoid such situations. A plan must be "safe". While applying the plan to the initial state of the world, given its foreseeable evolution modeled by external events, no situation shall arise where the "Fail" fact appears.

We have extended our planning algorithm in order to fulfill such a requirement. During the graph expansion

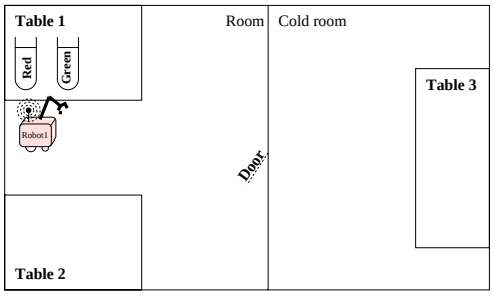

Figure 8: The frame's example

phase, the planner fires all events whose preconditions are satisfied. And, at each backtrack state the planner tests if in this partial state an event has possibly caused a failure. If so, the state is declared unsafe and the plan is rejected. This method ensures a safe plan synthesis even if an event appears after achieving the goal.

5 Results

In this section, we present our planner capabilities through a realistic example and sample problems. The current version is implemented in C++, and it uses an algorithm based in Graphplan.

As shown in figure 8, a robot stands beside $Table_1$ in a room which contains two tables ($Table_1$ and $Table_2$). On each table a lamp is fixed: the first lamp often lights whereas the second one usually lights. At the initial state, a red test tube and a green one are on $Table_1$. The goal is to cool the red test tube. A closed door separates the room from a cold-room in which there is a third table ($Table_3$). In order to cool a test tube, the robot must put it on $Table_3$. The robot may move from a location to another, take a test tube, see what it holds, open and close the door and put a test tube on a table. Table 1 shows the initial state.

First, we require a plan which achieves the goal with $N \geq 0.7$: $(\Pi(On(RedTestTube,\ Table_3)) = 1 \wedge N(On(RedTestTube,\ Table_3)) \geq 0.7)$. The planner finds a solution with 6 actions in 95 msec:

Take($RedTestTube$,$Table_1$) Move($Table_1$, $RoomDoor$)
Open($Door$) Move($RoomDoor$, $ColdRoomDoor$)
Move($ColdRoomDoor$,$Table_3$) Put($RedTestTube$,$Table_3$)

An advantage of our planner is its capability of finding a better (but more complex) plan upon request; for example if its requested to find a plan that satisfies the same goal with higher necessity ($N \geq 0.9$), the planner produces:

Take($Table_1$) Move($Table_1$,$Table_2$) WhichOne?($Table_2$)
if Robot holds RedTestTube
then Move($Table_2$,$RoomDoor$) Open($Door$)
Move($RoomDoor$, $ColdRoomDoor$)
Move($ColdRoomDoor$,$Table_3$) Put($RedTestTube$,$Table_3$)
if Robot holds GreenTestTube
then Put($GreenTestTube$,$Table_2$) Move($Table_2$,$Table_1$)
Take($RedTestTube$,$Table_1$) Move($Table_1$,$RoomDoor$)
Open($Door$) Move($RoomDoor$, $ColdRoomDoor$)
Move($ColdRoomDoor$,$Table_3$) Put($RedTestTube$,$Table_3$)

In this case, the planner has been obliged to use non-deterministic action as well as perception action to know which test tube it holds. Note that, the light of $Table_1$ does not allow the robot to see the tube that it holds, and thus the robot must go to $Table_2$ so as to distinguish its color. In addition, we observe that the first three actions define a conformant plan [Smith and Weld, 1998].

In the third example, we add two external events: if the cold-room door stays open too long (10 levels), the contents of the cold-room will be damaged; if the robot stays too long in the cold-room with the door closed it will be damaged. The planner finds a safe conditional plan (in 11780 msec) that satisfies $(N(On(RedTestTube, Table_3) \wedge \neg damage) \geq 0.9)$:

Take($Table_1$) Move($Table_1$,$Table_2$) WhichOne?($Table_2$)
if Robot holds RedTestTube then
Move($Table_2$,$RoomDoor$) Open($Door$)
Move($RoomDoor$,$ColdRoomDoor$) Move($ColdRoomDoor$,$Table_3$)
Put($RedTestTube$,$Table_3$) Move($Table_3$,$ColdRoomDoor$)
Move($ColdRoomDoor$, $RoomDoor$) Close($Door$)
if Robot holds GreenTestTube then
Put($GreenTestTube$,$Table_2$) Move($Table_2$,$Table_1$)
Take($RedTestTube$,$Table_1$) Move($Table_1$,$RoomDoor$)
Open($Door$) Move($RoomDoor$, $ColdRoomDoor$)
Move($ColdRoomDoor$,$Table_3$) Put($RedTestTube$,$Table_3$)
Move($Table_3$,$ColdRoomDoor$) Move($ColdRoomDoor$, $RoomDoor$)
Close($Door$)

Note that the door is only closed at the end of the plan: close the door is not a reflex, it has been planned and inserted in the right place.

To upgrade our goal description, we will introduce, in further work, the notion of flexible goals as in POSPLAN [Da Costa Pereira *et al.*, 1997] with qualitative utility [Dubois and Prade, 1995]. Indeed the robot security can be more important than mission achieving, that we will model by $N(On(RedTestTube, Table_3) \geq 0.9 \wedge N(cold) = 1$.

In table 2, we present set of a classical problems solved by our planner. Note that the CPU time is not completely significant, because the current version was implemented only to validate our representation. "Bomb in Toilet" is a problem proposed by [McDermott, 1987] (with sensing action), "Medical" by [Weld *et al.*, 1998] and "Fetch another package" by [Pryor and Collins, 1996]).

Problems	ND nodes	Nodes	CPU (msec)
ColdRoom	4	97342	11780
Bomb in Toilet	2	38	30
Medical	4	32	10
Fetch-2	4	212	80

Table 2: Some problem results (On an Ultra-Sparc 10)

6 Conclusion

In this paper, we have represented the uncertainty with the possibility theory; this allowed us to distinguish between two types of non-determinism: a non-determinism from insufficient modeling and a non-determinism from

uncertainty. Besides, we have introduced perception actions as well as a model of the environment dynamics through "contingent events". Finally, we have presented an implemented experimental planner which is able to produce plans that are robust with respect to contingent events, and whose goal-achieving ability is evaluated a priori. The obtained plans can be conformant or conditional depending on the context and the user requirements.

6.1 Related work

WARPLAN [Warren, 1976] is the first planner which works on the non-deterministic action and conditional effects. We can distinguish two planner types: the robot environment is uncertain and safe. For example BURIDAN [Kushmeric *et al.*, 1995] is a conditional probabilistic planner. The plan produced by BURIDAN is a linear action sequence which achieves the goal whatever the initial world. POSPLAN [Da Costa Pereira *et al.*, 1997] is possibilistic planner which searches an optimal plan with uncertainty non-determinism. UWL [Etzioni *et al.*, 1992] introduces a formalism for conditional branches and observations with runtime variables. C-BURIDAN [Draper *et al.*, 1994] proposes perception actions with probabilistic outcomes (noisy sensors). The complete ignorance modeling is the main problem of these planners because of the probabilistic representation. CASSANDRA [Pryor and Collins, 1996] is a non-deterministic planner with conditional effects which can only model the complete ignorance. SGP [Weld *et al.*, 1998] is the most recent non-deterministic planner with complete ignorance modeling. It is an extension of the conformant planner CGP[Smith and Weld, 1998] with sensing actions. SGP builds quickly plans using Graphplan algorithm. We can note that SGP cannot produce new worlds during planning; moreover it can plan without sensing actions if the world is unobservable. The second type of planners deals with on dynamical environments as [Kabanza, 1992] works about reactive planning. The environment is uncertain and dynamic, but observable. It has defined the safe situation notion to have secure planning.

6.2 Future work

Our planner can build safe conditional plans involving non-deterministic actions and perception actions. However, it does not have the ability to concentrate on the exploration of the most "probable" courses of action and to anticipate the (safe) situations where it will have to replan if it detects an error during execution. As we mentioned it previously, we will also integrate "flexible" goals which will allow to model the robot safety as more important than the mission. Another direction is to study more elaborate treatment of contingent events based on temporal reasoning.

References

[Anderson *et al.*, 1998] C. Anderson, D. Smith, and D. Weld. Conditional effects in graphplan. In *Proc. 4th Int. Conf. AI Planning Systems*, 1998.

[Blum and Furst, 1997] A.L. Blum and M.L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, pages 281–300, 1997.

[Da Costa Pereira *et al.*, 1997] C. Da Costa Pereira, F. Garcia, J. Lang, and R. Martin-Clouaire. Possibilistic planning: Representation and complexity. In *4th European Conference on Planning (ECP'97)*, 1997.

[Draper *et al.*, 1994] D. Draper, S. Hanks, and D. Weld. Probabilistic planning with information gathering and contingent execution. pages 31–36, 1994.

[Dubois and Prade, 1988] D. Dubois and H. Prade. Possibility theory - an approach to computerized processing of uncertainty. *Plenum Press*, 1988.

[Dubois and Prade, 1995] D. Dubois and H. Prade. Possibility theory as a basis for qualitative decision theory. *IJCAI'95*, pages 19–25, 1995.

[Etzioni *et al.*, 1992] O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson. An approach to planning with incomplete information. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, 1992.

[Fikes and Nilsson, 1971] R.E. Fikes and N.L. Nilsson. Strips: A new approach to the application of therom proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Kabanza, 1992] F. Kabanza. Reactive planning of immediate actions. *PhD thesis, Universite de Liege*, 1992.

[Koehler *et al.*, 1997] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an adl subset. In *4th European Conference on Planning (ECP'97)*, 1997.

[Kushmeric *et al.*, 1995] N. Kushmeric, S. Hanks, and D. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 2:239–286, 1995.

[McDermott, 1987] D. McDermott. A critique of pure reason. *Computational Intelligence*, 3:151–160, 1987.

[Perret and Alami, 1995] J. Perret and R. Alami. Planning with non-deterministic events for enhanced robot autonomy. In *4th International Conference on Intelligent Autonomous Systems (IAS-4)*, 1995.

[Pryor and Collins, 1996] L. Pryor and G. Collins. Planning for contingencies: A decision-based approach. *Artificial Intelligence Research*, 1996.

[Smith and Weld, 1998] D. Smith and D. Weld. Conformant graphplan. In *Proc. 15th Nat. Conf. AI.*, 1998.

[Warren, 1976] D. Warren. Generating conditional plans and programs. In *Proceedings of AISB Summer Conference*, pages 344–354, 1976.

[Weld *et al.*, 1998] D. Weld, C. Anderson, and D. Smith. Extending graphplan to handle uncertainty and sensing actions. In *Proc. 15th Nat. Conf. AI.*, 1998.

[Zadeh, 1978] L.A. Zadeh. Fuzzy sets as a basis for theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.