Acting in Partially Observable Environments When Achievement of the Goal Cannot be Guaranteed

Alexandre Albore
Universitat Pompeu Fabra
Barcelona, Spain
alexandre.albore@upf.edu

Héctor Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain
hector.geffner@upf.edu

Abstract

The problem of planning in partially observable environments can be regarded as a search problem in belief space where beliefs express the collection of states that are deemed possible. In this paper we address the problem that arises when one of the possible states is a dead-end: a state from which the goal cannot be reached. In such situations, no contingent plan exists, and yet, such situations are common when planning with incomplete information. For example, if a robot has to move to a target while sensing if adjacent cells are free or not, and the map is not known, the (hidden) state where the target is totally blocked by occupied cells is possible and is a dead end. Of course, the robot shouldn't get paralysed in such cases; it should move toward the target and give up only when one such state is not only possible but certain. However, contingent planning doesn't help in obtaining such behavior, and neither does the introduction of probabilities. A belief state where a dead-end state is possible is itself a dead-end, and both its worst case cost and its expected cost are infinite. One option in such cases is to find contingent plans or policies that maximize 'coverage', i.e. the set of possible states for which the solution works. This is the approach that we take in this paper, where we extend a recent action-selection mechanism for contingent planning, that uses a translation into classical planning, to work in such settings. We show that such scenarios are common and that the proposed mechanism has other applications as well. In particular, it can be used to deal with dead-end beliefs that do not contain dead-end states, and to generate meaningful, goal-oriented behavior in solvable but complex contingent settings where state-of-the-art contingent planners fail.

Introduction

Consider the problem of a robot that has to move from one position to another position in a grid, not knowing which of the cells in the grid are free. The robot can sense which adjacent cells are free and can move into an adjacent free location. The problem can be expressed as a contingent planning task and fed easily into a state-of-the-art contingent planner such as Contingent FF (Hoffmann and Brafman 2005), MBP (Bertoli et al. 2001), or POND (Bryce, Kambhampati, and Smith 2006). The planners, however, will not help, because the problem has no solution. Indeed, there are 'contingencies' in the problem that prevent the agent from reaching its destination with certainty; namely, the possibilities where all the paths to the goal are blocked. Still, in the absence of contingent plans, we do not want the robot to freeze. What it should do instead is to move toward the target and give up only when one of these possibilities turns out to be true. Contingent planners, however, do not capture such behavior. This limitation, all the same, does not apply only to contingent planning, but to many other models of action selection that presume that the goal can be achieved with certainty. Indeed, the problem can be cast as a POMDP by filling in the probabilities that each cell is free and maintaining the goal of reaching the target with certainty (Cassandra, Kaelbling, and Littman 1994). Yet even then, the expected cost of reaching this target belief will be infinity as there is a non-zero probability that the paths to the goal are blocked.[1]

Since contingent and POMDP problems can be cast as search problems in *belief space* (Astrom 1965; Bonet and Geffner 2000), the problems above correspond to situations in which certain belief states are *dead-ends*; belief states from which the goal beliefs cannot be guaranteed to be reached with certainty. The belief states in the contingent setting represent the set of states that the agent deems possible at one point. In this example, the states encode the position of the agent and the status of the cells (free or not), thus the initial belief state is a dead-end because it contains states where the non-empty cells block the paths to the goal. Those *states* are dead-ends in themselves as the goal cannot be reached from them even if such states are assumed to be observable. In contingent planning or in POMDPs, a belief state where a dead-end state is possible, is itself a dead-end from which the goal cannot be reached.

When planning with incomplete information, dead-end beliefs do not arise only from dead-end states. In a medical problem, for example, where solving depends on exactly identifying the patient's disease, if the observations available discard all but two possible diseases, and no therapy works for both of them, the belief is a dead-end (no pun intended) as the goal cannot be reached with certainty, even if each of

[1] Costs and rewards are often discounted in POMDPs, and that results in expected costs and rewards to be bounded. Yet, while discounting bounds costs, it does not produce meaningful policies when reachability cannot be ensured.

the diseases could be treated separately.

In the presence of dead-end beliefs, where the achievement of the goal cannot be guaranteed, one option is to find contingent plans or policies that maximize 'coverage', i.e. the set of possible states for which the solution works. Such policies are well-defined except in the very unfortunate situations where each of the states that are deemed possible is a dead-end. None of the scenarios above, however, falls into this class. This is the approach that we take in this paper, where we extend a recent *action selection mechanism* for contingent problems to work in such settings. We show through a number of examples that such scenarios are common and the proposed extension has other applications as well. In particular, it can be used to generate meaningful, goal-oriented behavior in solvable but complex contingent settings where state-of-the-art planners fail.

The action selection mechanism on which we build is the one formulated for the CLG planner: a contingent planner that can work in off-line mode, for building full contingent plans, or in on-line mode, for generating single executions (Albore, Palacios, and Geffner 2009). In execution mode, CLG uses the model $X(P)$ to keep track of the beliefs, and a relaxation of $X(P)$ for selecting the actions. CLG relies on a translation that maps a non-deterministic problem in belief space P (a contingent planning problem), into a non-deterministic problem $X(P)$ in state space, where beliefs are represented by means of new literals KL/t that express epistemic conditionals: namely, that if t is true initially, then L is true.

The advantages of building on CLG are two. First, in many of the problems, the bottleneck is not the lack of solutions, but the *size of the solutions*, that is exponential in the number of possible observations. The ability of CLG to produce goal-oriented observation/action sequences without having to build a complete contingent plan first, is thus a clear plus. Second, the use of the tags t in the underlying translation, that denote assumptions about the initial situation, can be used to find contingent plans or policies that (heuristically) maximize 'coverage'. This will be done by extending the translation $X(P)$ with actions that manipulate these assumptions.

The changes result in an action selection mechanism that we call CLG+, whose difference with CLG is that it works in contingent problems without solutions. Thus CLG+, like CLG, does not 'freeze' due to the size of the contingent solutions, but unlike CLG, it does not 'freeze' either when such solutions do not exist. As long as there is the possibility of reaching the goal, CLG+ will go for it, making CLG+ a quite robust and persistent action selection mechanism. We will illustrate its behavior through a number of examples, many of which fall outside the scope of the planners and solvers that we are aware of.

The paper is organized as follows. We first review contingent planning (Section 2), the problem of dead-ends in belief space (Section 3), and the CLG planner (Section 4). We then consider the extensions that make CLG+ (Section 5), and illustrate its behavior over a number of examples (Section 6). We then evaluate the overhead of CLG+ in relation to CLG (Section 7), and end with a summary.

Contingent Planning

The contingent planning problems that we consider depart from classical planning problems into two ways: first, the initial situation may be uncertain, second some of the actions are sensing actions. The contingent problems are thus tuples $P = \langle F, O, I, G\rangle$ where F stands for the fluent symbols in the problem, O stands for the actions set, I for a set of *clauses* over F defining the initial situation, and G for a set of literals over F defining the goal. All the actions are assumed to be deterministic and all uncertainty is assumed to lie in the initial situation only.

A normal action a has a precondition given by a set of fluent literals, and a set of conditional effects $C \rightarrow L$ where C is a set of fluent literals and L is a literal. A literal is a fluent or its complement, and the expression $\neg L$ is used to denote the complement of a literal L .

The sensing actions uncover the truth value of a fluent x in F and are denoted as $obs(x)$. Sensing actions can have preconditions as any other actions, but for simplicity we assume that they have no other effects.

The solution to a contingent problem P can be expressed as a contingent plan or tree where the branches capture the possible executions. A plan solves the problem if all the possible executions are feasible (action preconditions hold when actions are applied) and end up in goal states (Bertoli et al. 2001; Hoffmann and Brafman 2005; Bryce, Kambhampati, and Smith 2006).

An alternative characterization, useful for our purposes, can be obtained from a dynamic programming formulation (Bellman 1957; Bonet and Geffner 2000). In a contingent problem, an agent starts in a given belief state $b = b_0$ and must reach a target belief b_F containing only goal states. For this, the *physical effects* of an action a, applicable in a belief state b, deterministically map b into a new belief state b_a:

$$b_a = \{s' \,|\, s' = f(a, s)\,,\; s \in b\}$$

where $f(a, s)$ is the state-transition function determined by the effects associated with the action a in P, while the *sensing effects* of a non-deterministically map b_a into a belief state b_a^o:

$$b_a^o = \{s \,|\, s \in b_a \text{ and } s \text{ compatible with } o\}$$

where o is one of the possible observations that may arise in b_a. If we write $O(b, a)$ to indicate the possible observations that may arise by doing action a in b, then the cost of reaching a target belief state b_F from a non-target belief b, in the worst case, can be obtained from the optimality equation:

$$V(b) = \min_{a \in A(b)} \Big[c(a) + \max_{o \in O(b,a)} V(b_a^o)\Big]$$

with $V(b_F) = 0$ for goal beliefs (those that include goal states only). $A(b)$ denotes the set of actions applicable in b; namely, those whose preconditions are true in b, and $c(a)$ is the cost of the action a, assumed to be 1 by default.

The solution to the optimality equation yields the optimal cost function $V^*(b)$ that measures the cost of reaching a goal belief from b in the worst case. If action costs are all 1, $V^*(b)$ provides the depth of the contingent tree

with minimal depth that solves the problem. In our setting, where actions have either physical or sensing effects but not both, $O(b, a)$ must be set to contain just one dummy observation that is true in all states when a is a physical action, and b_a must be set to b when a is a sensing action. For capturing expected costs rather than worst case costs, the beliefs must be set to probability distributions, and the equations for updating beliefs and the optimality equations must be adjusted, with a weighted sum over the observations $o \in O(b, a)$ replacing the max. The resulting model is a POMDP (Cassandra, Kaelbling, and Littman 1994; Bonet and Geffner 2000).

A dead-end belief in this setting is just a belief state b with infinite cost $V^*(b)$, reflecting that a goal belief cannot be reached with certainty from b. Problems where the initial belief state b_0 is a dead-end have no solution, and neither contingent nor POMDP planners yield a behavior for them, which is certainly a weakness of those formulations, as such situations are common.

A dead-end state s is a state from which the goal cannot be reached even supposing full observability. Such states have infinite optimal costs $V^*(s)$ and they induce an infinite optimal cost on the beliefs states that render them possible. Indeed, it is easy to show that $V(b) \geq V^*(s) = \infty$, either in the contingent or POMDP settings.

Yet action should not be stopped when one of the possible scenarios is a dead-end but rather when these possibilities are the only ones left. A possible principle for guiding such an action is by following a policy obtained from a relaxation where the current belief is augmented with assumptions. These assumptions can be then confirmed or disconfirmed as a result of the observations gathered. This process can be iterated until reaching the goal or finding out that the goal is unreachable. This is the approach taken in (Albore and Bertoli 2006) where the assumptions to make are given explicitly. In our framework, they will arise naturally from the planning process.

Dead-Ends in Belief Space

Figure 1 shows an instance of the example used in the introduction: a 5×5 grid where the cells in the middle column, shown in gray, may be free or not, and an agent that is initially to the left of this column must move to the other side. The status of those cells is not known to the agent, but the agent can move one unit in each of the four directions, if the corresponding destination cell is free, and can sense the status of the adjacent cells.

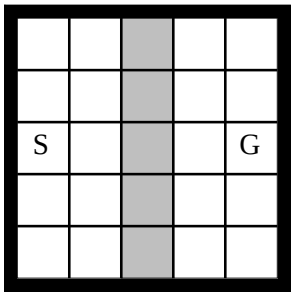


Figure 1: A 5×5 grid.

The problem involves in the order of $5^2 \times 2^5$ states: 5^2 possible positions for the agent, and 2^5 possible configurations of the cells in gray. Some of these states are not reachable, such as those where the agent is at cell that is not free. The dead-end states s for this problem are quite few and correspond to those in which the agent is on the left and none of the cells in the middle column is free. These are 10 states, and one of these 10 states is part of the initial belief state b_0, and hence b_0 is a dead-end belief, and the problem has no solution.

We fed this problem into some recent contingent planners. Contingent-FF, for example, notices right away, without search, that the problem has no solution and immediately quits. POND, on the other hand, starts searching for a solution and keeps searching for quite a while. CLG behaves like Contingent-FF, producing an infinite heuristic value for the initial belief state, quitting right away without doing any search. In this case, the responses of both Contingent-FF and CLG are adequate if the objective is the generation of a plan that must reach the goal with certainty considering all the possibilities. However, this is not a reasonable objective nor a requisite for an agent acting in partially observable environments. Actually, CLG can be used in on-line mode where it does not build a full contingent plan before the next action is executed. In such mode, CLG becomes an action selection mechanism that outputs the next action to do given the past observation-action sequence. However, the heuristic used in CLG for producing this behavior implicitly considers all the possibilities, something that makes perfect sense if the choice is between an action that works in all cases, and one that works only on some, but fails when the choice is between actions none of which works in all cases.

In the absence of a policy that works in all cases, the strategy that makes sense is to follow a policy that will work in most cases, revising the policy as new observations are gathered. This means actually to compute a policy assuming the current belief not to be b, if b is a dead-end, but a belief b' that differs minimally from b and is not a dead-end. In the example depicted in this figure, this b'_0 can be obtained by excluding from b_0 the states where all gray cells are blocked, or more simply, by keeping only the states in which one specific grey cell is free. Such a change amounts to making an assumption. If the assumption is found out to be wrong, it can then be revised and replaced by another assumption if the resulting belief state is still a dead-end.

This is the strategy that we will adopt, that fits naturally with the CLG planner, which is built on a translation where the assumptions t about the initial situation are part of the language. The key idea will be to extend the translation with actions that manipulate these assumptions.

CLG: Closed Greedy Planner

CLG is an action selection mechanism for acting in partially observable environments that can be used in on-line mode, for selecting the next-to-apply action on a single execution given the past sequence of observations and actions, or in off-line mode, for building full contingent plans (Albore, Palacios, and Geffner 2007; 2009). For the latter task, it has been shown to scale up better than other recent planners such as Contingent-FF and POND, while its on-line mode allows it to deal with problems with too many contingencies where the construction of full contingent plans is not feasible. The action selection mechanism that we propose for

dealing with contingent scenarios with no solutions is based on CLG and exploits its formulation.

Translation

CLG is based on a translation that compiles beliefs away: contingent problems P are translated into non-deterministic but fully observable problems $X(P)$ whose literals encode the beliefs over P.

The translation $X(P) = X_{T,M}(P)$ involves two parameters, a set of *tags* T and a set of *merges* M, and builds on a similar translation $K_{T,M}$ introduced before for mapping conformant problems into classical problems (Palacios and Geffner 2007). In these translations, a tag t is a set (conjunction) of literals in P whose status in the initial situation I is not known, and a merge $m \in M$ is a collection of tags $t_1, \ldots, t_n$ that stands for the DNF formula $t_1 \vee \cdots \vee t_n$. Tags are assumed to represent consistent assumptions about I, i.e. $I \not\models \neg t$, and merges, disjunction of assumptions that are valid in I; i.e. $I \models t_1 \vee \cdots \vee t_n$.

The fluents in both $X_{T,M}(P)$ and $K_{T,M}(P)$, for $P = \langle F, O, I, G\rangle$ are of the form KL/t for each $L \in F$ and $t \in T$, meaning that "if t is true in the initial situation, L is true". In addition, $K_{T,M}(P)$ includes extra actions, called *merge actions*, that allow the derivation of a literal KL when KL/t' has been obtained for each tag t' in a merge $m \in M$.

The translation $K_{T,M}$ maps a conformant problem $P = \langle F,O,I,G\rangle$, into the *classical problem* $K_{T,M}(P) = \langle F',O',I',G'\rangle$ where

$$
\begin{aligned}
F' =& \{KL/t, K\neg L/t \mid L \in F\} \\
I' =& \{KL/t \mid \text{if } I \models t \supset L\} \\
G' =& \{KL \mid L \in G\} \\
O' =& \{a : KC/t \rightarrow KL/t,\ a : \neg K\neg C/t \rightarrow \neg K\neg L/t \\
& \mid a : C \rightarrow L \text{ in } P\} \cup \{\bigwedge_{t\in m} KL/t \rightarrow KL \mid m \in M\}
\end{aligned}
$$

with t ranging over T and the preconditions of the actions a in $K_{T,M}(P)$ including the literal KL if the preconditions of a in P include the literal L.

The expressions KC/t and $\neg K\neg C/t$ when $C = L_1, \ldots, L_n$, are abbreviations for $KL_1/t, \ldots, KL_n/t$ and $\neg K\neg L_1/t, \ldots, \neg K\neg L_n/t$ respectively. Similarly, KL stands for KL/t_0 where t_0 is the "empty tag". The empty tag is assumed in all sets T.

The translation $K_{T,M}(P)$ is sound, and for suitable choices of tags and merges is *complete*, the former meaning that the classical plans that solve $K_{T,M}(P)$ yield valid conformant plans for P (by dropping the merge actions), and the latter that all conformant plans for P can be obtained in this way.

The translation $X(P) = X_{T,M}(P)$ used in CLG is the translation $K_{T,M}(P')$ of the *conformant fragment* P' of P (i.e. P without the sensing actions) extended with two components: an encoding of the *sensing actions*, expressed as non-deterministic actions, and two *deductive rules* expressed as actions that extend the conformant merges. For suitable choices of tags and merges, the translation $X_{T,M}(P)$ is complete too, and a similar family of translations $X_i(P)$ that are complete for problems with *contingent width* no greater than i is defined in (Albore, Palacios, and Geffner 2009).

The sensing actions $obs(L)$ in P become in $X(P)$ the *non-deterministic actions*

$$obs(L) : \neg KL \wedge \neg K\neg L \ \rightarrow\ KL \,|\, K\neg L\,,$$

while the deductive rules, encoded as actions with single conditional effects are:

$$
\begin{aligned}
&\bigwedge_{t\in m, m\in M} (KL/t \vee K\neg t) \ \rightarrow\ KL \\
&KL/t \wedge K\neg L \ \rightarrow\ K\neg t
\end{aligned}
$$

The computational pay-off of the translation $X(P)$ is that it encodes beliefs by sets of literals. The translation $X(P)$ is solved in CLG by using a relaxation $H(P)$ that is called the heuristic model. This model represents a classical planning problem obtained from P by moving preconditions in as conditions, discarding deletes, and encoding sensing actions as deterministic actions over literals ML that express contingent knowledge and are used to encode action preconditions.

Action Selection

The *Closed-Loop Greedy (CLG)* planner uses the execution model $X(P) = X_1(P)$ for keeping track of beliefs, and the heuristic model $H(P)$ for selecting the actions to do next in closed-loop fashion. Starting with the initial state s of $X(P) = X_1(P)$, an action sequence π is selected for application in s, and the loop resumes from the state s' that results, until s' is a goal state in $X(P)$. The action sequence π is obtained using a modified version of the classical FF planner. In FF, a single enforced hill climbing (EHC) step is a local search that results in an action sequence π that maps a state s into a state s' with a better heuristic value h_{FF}. In this local search, the classical model is used for doing the state progression, and its delete-relaxation for computing the relaxed plans. CLG adopts the same search strategy, with the execution model $X(P)$ used for the progression, and the heuristic model $H(P)$ used for computing the relaxed plans. In addition, in order to avoid the consideration of non-deterministic actions in the local search, whenever a "local plan" π that ends in a sensing action $obs(L)$ is being considered, the action sequence π is returned without further evaluation. Notice that for an action to be considered into the local plan, the action must have been found to be "helpful" according to FF's criterion.

CLG+

The modifications needed to render CLG usable in contingent problems where the achievement of the goal cannot be guaranteed, makes use of two main ideas and opens the scope of the resulting action selection mechanism quite considerably.

The key innovation in CLG+ is the introduction of *assumptions* in the form of new actions with effect $K\neg t$ and precondition $\neg K\neg t$ and $\neg Kt$. We will abuse notation

slightly and name those actions as $K\neg t$. These actions form part of both the execution model $X(P)$ and the heuristic model $H(P)$. The reason for having assumptions of the form $K\neg t$, rather than of the form Kt, is that the former are finer grained. For example, in a problem involving an initial situation with an $xor(x_1, \ldots, x_n)$, where the tags are $t_i = x_i$, we will be able to discard some x_i without necessarily committing to another one.

The second element needed is the handling of action costs. In order to establish a preference for solutions and relaxed solutions that work in most cases, actions that involve assumptions, while feasible, are penalized with a high cost. Then, an heuristic sensitive to costs is needed for selecting the actions. In order to preserve as much of the architecture of CLG as possible, we achieve this in CLG+ by moving from FF, as the underlying classical planner, to FF(h^s_a), a planner that retains from FF everything except the definition and computation of the relaxed plans (Keyder and Geffner 2008). In FF, the relaxed plans that serve to provide the heuristics and the helpful actions are computed using the relaxed plan graph. This computation, however, is not sensitive to costs. In FF(h^s_a), the relaxed plans are computed recursively, very much as the heuristic estimates in the additive heuristic. The difference between the additive heuristic h_{add} and the set-additive heuristic h^s_a, is that the former propagates numbers $h(p; s)$ that estimate the cost of achieving the fluent p from s, while the latter propagates labels $\pi(p; s)$ that capture the relaxed plans for computing p from s. While cost-sensitive relaxed plans could be obtained from using the least expensive additive heuristic (Keyder and Geffner 2008), the set-additive heuristic has benefits when dealing with conditional effects, that are numerous in the translation $H(P)$ and are not treated well by most heuristics. Indeed, the h_{FF} heuristic treats conditional effects as independent actions, except when the conditional effects appear in the same level of the relaxed plan graph. This choice is rather arbitrary, as indeed, sometimes the heuristic value of a state can be reduced by moving a conditional effect to a successive layer. The set-additive heuristic combines for us a sensitivity to costs with the ability to deal with conditional effects in a more principled manner. Taking advantage of a cost-sensitive heuristic, the cost of all the deductive actions in $X(P)$ and $H(P)$ is set to 0.

A third element in the move from CLG to CLG+ is a device for preventing inconsistent assumptions from producing arbitrary conclusions. While the deductive actions in CLG are not deductively closed, the merge rules in the heuristic model may result in unwarranted conclusions. In particular, if m is a merge for the literal L, the contingent merge action $\bigwedge_{t\in m}(KL/t \vee K\neg t) \rightarrow KL$ can be used to derive KL from the assumptions $K\neg t$ for all $t \in m$. These assumptions are jointly inconsistent, but nothing prevents them from being made in a 'relaxed plan' with a sufficiently high cost. In order to avoid this, we introduce a new fluent in the language, $ok(m, L)$ that we keep true only when *some* literal KL/t is true, and introduce this new literal as an extra condition in the merge action. This prevents the merge m for L to trigger if none of the literals KL/t are true.

Finally, a last issue that needs to be addressed in CLG+, is the difference between assumptions $K\neg t$ in the execution model $X(P)$ and the same assumptions in the heuristic model $H(P)$. Recall, that the heuristic model is used to compute heuristic values and relaxed plans, and to indicate the actions that are helpful. On the other hand, states are progressed using the execution model, both when they are finally selected and applied, and when they are applied in the EHC search. The difference between an assumption $K\neg t$ in $H(P)$ and the same assumption in $X(P)$ is that the former is just 'thinking', while the latter is 'acting'; or in different words, the first just extends the relaxation with an extra assumption, the second is a commitment that affects the current beliefs. As we will see, sometimes the current beliefs need to be changed, as for example, when there is a single possible relevant action to make, but we are uncertain about the status of its preconditions. If the agent is supposed to head for the goal no matter what, as CLG+ does, then it must then take risks: if it applies an action assuming that its preconditions hold in the current true but hidden state, and this is not the case, then the execution will fail. Otherwise, it's not a good idea to risk the execution. The asymmetry between assumptions $K\neg t$ in the execution and heuristic models is captured as follows. The actions to be done in the current state s are computed first, doing an EHC search from s, with the helpful actions but excluding the execution of assumptions (assumptions in $X(P)$). If this fails, then this EHC is repeated, considering all actions and not only the helpful ones, but still without executing any assumption. If this fails too, then this same EHC search is carried out with all the actions enabled, including the assumptions in the execution model. It is only then that assumptions may be chosen for execution. In all cases, as in CLG, if a sensing action is selected for application in the EHC search, the path leading to the sensing action is selected for application. As in CLG, this is done in order to avoid the simulation of the non-deterministic effects of sensing in the EHC search.

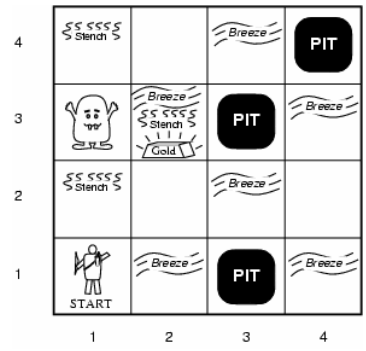
Examples

We illustrate the behavior of CLG+ over a number of different problems. Most are meaningful contingent problems without solutions. We include also problems with contingent widths greater than 1 which are not solved by CLG, but that CLG+ manages to solve. We divide the problems into those with dead-ends that arise from dead-end states, those with dead-ends that arise due to the absence of policies able to handle all possibilities, and those that are solvable but complex.

Problems with Dead-End States

Figure 2 shows a version of the Wumpus problem from (Russell and Norvig 1994). In this problem, the agent shown in the Start state must get the Gold by sorting dangerous wumpuses and pits. We assume that the agent can move deterministically one unit in each of the four directions, provided that in the target cell there is no wall, wumpus, or pit.[2]

[2]In the book, if the agent moves into a wumpus or pit cell, it dies. Also agent is then armed with an arrow which can kill the wumpus if fired properly, but does not know its position in the grid.




```
1: MOVE_P1-1_P2-1
2: MOVE_P2-1_P1-1
3: MOVE_P1-1_P1-2
4: MOVE_P1-2_P2-2
5: MOVE_P2-2_P3-2
6: MOVE_P3-2_P2-2
7: MOVE_P2-2_P2-3
8: GRAB_P2-3
```

Figure 2: An execution in Wumpus: hidden state shown on the left and execution (with passive sensing) shown on the right.

Initially, the agent knows its location, but doesn't know where the gold is, nor where the wumpuses or pits are. The uncertainty in the initial situation is modelled by having uncertainty in each cell regarding whether it contains a wumpus, a pit, none, or both. The uncertainty on the gold location is an XOR over all the grid cells. This is a natural encoding of the problem, yet it makes the problem unsolvable, as the gold may be in a cell with a wumpus or pit, or it may be blocked by them. The agent can sense the presence of a wumpus in *some* adjacent cell by the smell (stench), the presence of a pit in *some* adjacent cell by the breeze, and the presence of the gold in its own cell by the glitter. The stench, the breeze, and the glitter are all observable.

The figure produces the execution obtained from CLG+ by running it on the hidden state shown in the picture, where there are 3 pits and 1 wumpus as shown, and the gold is located at cell 2-3. For simplicity, the sensing is assumed to be passive, meaning that all applicable sensing actions are applied after every step. Thus, the actions shown in the execution all refer to non-sensing actions.

In the execution, the agent starts at 1-1 knowing that there is no wumpus or pit in the two adjacent locations. It then moves to 2-1, where it senses no stench, and thus, that there is not wumpus around, and a breeze, from which it concludes that the only safe cell to move is 1-1. It moves there, and from there it moves to the other known safe cell 1-2, where it senses no breeze, and thus no pit around, but it senses a stench. From the observations gathered so far it concludes that there is neither a wumpus nor a pit at 2-2, and thus that the cell is safe, and its moves there, where it senses no stench and no breeze. It decides to move then to 3-2, but sensing a breeze there, it backs up to 2-2, heading up now into 2-3, where it sees the glitter and grabs the gold.

In all this execution, CLG+ never commits to an action $K\neg t$ (an assumption) in the execution, but uses many of those actions in the construction of relaxed plans, and in the computation of the heuristic. Indeed, in every relaxed plan computed, it is assumed that the gold is not at a number of locations, from which it is inferred that it must be at particular location. This is not enough though. All the relaxed plans involve also assumptions about cells that are safe and need to be traversed in order to reach to the position in which the gold is suspected to be. All these assumptions are handled automatically as actions in the heuristic model. No assumptions from the execution model are done.

In this problem, the translations used in CLG+ took 1.5 seconds and the execution 5.5 seconds.

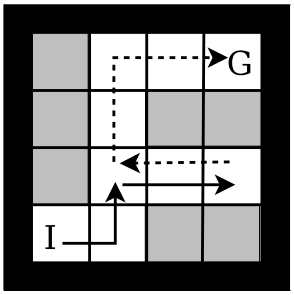


Figure 3: Navigation. Initially the status of all cells other than I and G is unknown. The hidden state is shown by indicating the free cells in white.

Navigation in Unknown Map We consider now a version of the problem used in the introduction, where an agent must move from an initial location I to a target location G in a grid where the status of all other cells is unknown (cf. Fig. 3). Such cells can be free or not, the agent can sense the status of adjacent cells, and can move into them if free. The problem is not solvable, as it contains dead-end beliefs that result from dead-end states: those in which the cells that are not free block the goal. Arrows in the figure indicate the execution that results from a particular hidden state where the cells that are free are the ones shown in white. In such a case, by moving and sensing, the agent follows the path shown. Not knowing that the two cells under the goal are blocked, the agent heads initially to the right, but having learned that, it backtracks, going around those two cells.

In this example, the translation took 0.3 seconds and the execution in CLG+ 0.5 seconds.

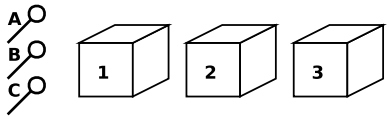


Figure 4: A boxes domain.

Learning Unknown Model We consider next a Boxes domain, illustrated in Fig. 4, where a treasure is hidden in one of three closed boxes. There are three levers, each controlling the opening or closure of one box, but the agent does not know which box. The problem is modeled with an uncertain initial situation where an XOR is used to state that each lever controls one box, but nothing implies that each box is controlled by one lever. Thus, the problem where the agent has to get the hidden treasure has no contingent solution. The actions used to open the boxes are the $push$ actions:

```
(:action push_lever
 :parameters (?l - lever)
 :effect (and (when (controls ?l box1)
                    (opened box1))
              (when (controls ?l box2)
                    (opened box2))
              (when (controls ?l box3)
                    (opened box3))  ))
```

Similar actions are used to closed them. The agent can check if the treasure is in a box if the box is open. For making the task more interesting, we also ask that the boxes be all closed at the end. This tests whether the agent is learning the action model; namely, which lever controls which box. Passive learning is assumed also in this case, meaning that all applicable sensing actions are automatically applied after each normal action. In this case, the observable fluents include whether a box is opened or closed, and whether the treasure is or is not in an open box.

An execution that is obtained for the hidden initial state where the treasure is in box 2, and the levers A, B, and C control the boxes 1, 2, and 3 respectively, is given by the action sequence: *push-a, push-b, pull-a, pick-treasure, pull-b*, where the sensing actions are applied after each step. As a result, after the first *push-a* action, the planner infers that level-a controls box 1, after *push-b*, it infers that the lever B controls the box 2 and that the treasure is there; it then closes box 1, picks up the treasure, and closes box-b, applying the knowledge gained about the lever mechanism.

In this example, the translation takes 0.04 seconds, and the execution in CLG+, 0.08 seconds.

Problems with Pure Dead-ends Beliefs

Here we illustrate an unsolvable problem that results from the absence of a policy that can deal with all the possible states in the initial belief state, even though, none of these states is a dead-end. Any policy that is taken will work for some initial states, but not for others. Thus, in these cases, the adoption of a policy involves 'betting': if we are lucky, we solve the problem, if we are not, we fail. This is different than in the problems above where the actions of the agent did not affect the solvability of the problem; the question then was whether the hidden state was a dead-end state or not.

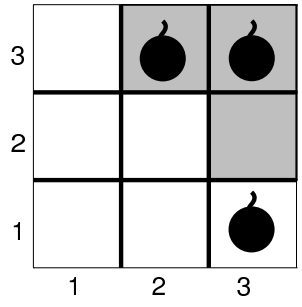


Figure 5: Minesweeper: distribution of bombs is the hidden state

Minesweeper We consider a version of the well known Minesweeper problem. This problem involves a grid where each cell may contain a mine or not. Cells without a mine must be cleared with the 'clear' action, while cells with a mine must be cleared with a 'sweepmine' action. The goal of the problem is to have all the cells cleared. Knowledge about the location of the mines is obtained through a 'check' action on a cell that yields a failed execution if the cell contains a mine. We model this by setting 'no mine at cell' as a precondition of 'check cell', and 'mine at cell' as precondition of 'sweepmine cell'. Given the initial complete lack of knowledge about the mine locations, the problem is unsolvable, and indeed, it can lead to failure after the first 'check'. One thus must 'bet' on the first cell to clear. The initial belief state is thus a dead-end even if does not contain any dead-end states. Indeed, the problem would be solvable for any initial state if the state was observable.

In the figure, the hidden state has mines at (2;3), (3;1), and (3;3), and the first action is to assume that there is no mine at (1,1), and to check this position. This is the first domain where $K\neg t$ actions need to be executed outside of the heuristic model, in order to obtain the knowledge about the action preconditions. This could have led to an execution failure if there was a bomb in (1,1). After the check at (1,1), for the hidden state shown, the planner infers that there is no bomb at either (1,2) or (2,1). It thus checks these two positions, and infers from the first check, that there is no mines at (1,3) and (2,2). On the other hand, from the check at (2,1) it infers that there must be a bomb at (3-1), since it is known by then that there is no bomb at (2,2), and minesweeps that bomb. The belief at that point is shown in the figure 5, where the white cells are the ones whose status is known. It then chooses to do a check at (3,1), that is now cleared, to find out that (2,3) is clear, and from the resulting observations finds out the status of the two other cells with bombs.

The instance is solved in 0.15 seconds by CLG+, with 0.13 seconds taken in the translation.

Problem with High Contingent Width

The last example is a solvable problem having contingent width higher than 1, that both CLG and Contingent-FF fail to solve, and where POND solves only the small instances. CLG+, on the other hand, turns out to scale up better in this example as a full contingent planner.

Binary Tree The problem involves a binary tree with depth n. Starting in the root, the agent has to move to a leave of the tree. The possible actions are to move from a node n to its left or to its right son, if the corresponding edge is not blocked. One of the two edges, however, is always blocked, and the agent can find out which by doing a sensing operation at n. There are also actions for moving up in the tree. The problem has width n, meaning that a complete $X_i(P)$ translation would be exponential in n. As a result, CLG that uses the $X_1(P)$ translation does not solve the problem, and reports an infinite heuristic value. The problem, however, is simple, just requiring sensing at each node, and moving to the son along the edge that is open. While CLG does not find a solution to this solvable problem, CLG+ does: it finds

a full solution to trees of depth 3, 4, and 5, in 0.24 seconds, 0.99 seconds, and 3.60 seconds. POND solves the first in 0.11 seconds, the second in 0.87 seconds, and does not solve the last one.

Scalability and Overhead

	CLG		CLG+	
problem	**time**	**#acts**	**time**	**#acts**
ebtcs-50	11.96	149	12.88	149
ebtcs-70	34.37	209	34.13	209
medpks-50	3.23	101	3.17	101
medpks-70	9.89	141	9.10	141
medpks-99	28.77	199	27.61	199
unix-3	9.26	113	52.17	111
unix-4	120.72	240	1748.84	238
cballs-4-1	0.35	295	0.71	282
cballs-4-2	18.83	20050	56.44	20203
cballs-4-3	1537.99	1136920	T	
cballs-9-1	192.16	3385	234.88	3497
cballs-9-2	T		T	
clog-7	0.17	210	1.12	215
clog-huge	157.94	37718	T	
doors-7	10.60	2153	64.28	2145
doors-9	1042.96	46024	T	
wumpus-5	1.76	732	14.31	753
wumpus-7	89.32	10681	1217.39	17256
wumpus-10	T		T	

Table 1: Examples to calibrate the performance and scalability of CLG+ in relation to plain CLG. Figures shown are total time and total number of actions in solution.
'T' stands for time out (cut-off of 45mn or 1.8Gb of memory).

Table 1 displays the ability of CLG+ to build *full contingent plans* over standard benchmarks, in comparison with CLG, shown in (Albore, Palacios, and Geffner 2009) to scale up better than Contingent-FF and POND. All the tests of this paper are obtained on a Linux machine running at 2.33GHz with 2Gb of RAM.

In these problems, all the additional machinery in CLG+ is not needed, and thus the difference in performance between CLG and CLG+ shows the overhead resulting from these changes. As it can be seen, CLG+ is slower than CLG, but it manages to solve most of the problems that CLG can solve. These figures give an idea of how well CLG+ scales, an idea that cannot be obtained from the examples discussed above, that are aimed at describing the functionality of CLG+ rather than its scalability.

Summary

We have presented an action selection mechanism for acting in partial observable environments where the achievement of the goal cannot be guaranteed. As we have seen, this is a common situation when planning with incomplete information, and yet, contingent and POMDP approaches, do not appear to capture the behavior that is sensible in those settings. The proposed action selection mechanism, called CLG+, that extends the one in the CLG planner, can deal with dead-end beliefs arising from dead-end states and situations where no strategy works in all cases, and can solve contingent problems, such as those with high contingent width, that cannot be solved by state-of-the-art planners. Like CLG planner, CLG+ does not get paralysed due the size of contingent solutions, because it does not have to build such solutions in order to act. At the same time, unlike CLG, CLG+ does not get paralysed either when such solutions do not exist. As long as there is the possibility of reaching the goal, CLG+ will go for it, making it into a robust and persistent action selection mechanism able to work in a wide variety of scenarios.

References

Albore, A., and Bertoli, P. 2006. Safe LTL assumption-based planning. In *Proc. 16th Int. Conf. on Planning and Scheduling (ICAPS-06)*.

Albore, A.; Palacios, H.; and Geffner, H. 2007. Fast and informed action selection for planning with sensing. In *Proc. 12th Conf. Spanish AI (CAEPIA-07)*. Springer.

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proc. 21st Int. Joint Conf. AI (IJCAI-09)*. Forthcoming.

Astrom, K. 1965. Optimal control of markov decision processes with incomplete state estimation. *J. Math. Anal. Appl.* 10:174–205.

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proc. 13th Int. Joint Conf. AI (IJCAI-01)*.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-2000*, 52–61. AAAI Press.

Bryce, D.; Kambhampati, S.; and Smith, D. E. 2006. Planning graph heuristics for belief space search. *Journal of AI Research* 26:35–99.

Cassandra, A.; Kaelbling, L.; and Littman, M. L. 1994. Acting optimally in partially observable stochastic domains. In *Proc. AAAI*, 1023–1028.

Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In *Proc. 15th Int. Conf. on Automated Planning and Scheduling (ICAPS-05)*.

Keyder, E., and Geffner, H. 2008. Heuristics for planning with action costs revisited. In *Proc. European Conf. AI (ECAI-08)*.

Palacios, H., and Geffner, H. 2007. From conformant into classical planning: Efficient translations that may be complete too. In *Proc. 17th Int. Conf. on Planning and Scheduling (ICAPS-07)*.

Russell, S., and Norvig, P. 1994. *Artificial Intelligence: A Modern Approach*. Prentice Hall.