Planning for partial observability by SAT and graph constraints

Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)

Planning for Partial Observability by SAT and Graph Constraints

Binda Pandey, Jussi Rintanen*
Department of Computer Science
Aalto University, Helsinki, Finland

Abstract

Chatterjee et al. have recently shown the utility of SAT in solving a class of planning problems with partial observability. A core component of their logical formulation of planning is constraints expressing s-t-reachability in directed graphs. In this work, we show that the scalability of the approach can be dramatically improved by using dedicated graph constraints, and that a far broader class of important planning problems can be expressed in terms of s-t-reachability and acyclicity constraints.

Introduction

Chatterjee et al. (2016) proposed a new framework for solving contingent planning problems with partial observability, bounded memory, and the objective of reaching a goal state. This planning problem can be understood as a simplification of the partially observable Markov decision process (POMDP) problem, in which transition probabilities are ignored, and instead of numeric rewards on all transitions, reaching a goal state is viewed as generating a reward.

Interestingly, this problem for a fixed number of memory states is NP-complete (Chatterjee, Chmelik, and Davies 2016). In contrast, classical planning is polynomial-time solvable in the size of the state space (by a simple graph traversal of the state space), and the POMDP problem is unsolvable in general (Madani, Hanks, and Condon 2003) and PSPACE-hard even when restricted to short horizons with lengths that equal the cardinality of the state space (Mundhenk et al. 2000; Papadimitriou and Tsitsiklis 1987). Hence the limitation to small-memory policies represents a dramatic reduction in the complexity of the problem. Further, and more interestingly, there is a relatively straightforward reduction to the satisfiability problem of the propositional logic, SAT. This provides an interesting addition to planning problems solvable by SAT and related methods (Kautz and Selman 1996; Cimatti et al. 2003; Shin and Davis 2005; Ferraris and Giunchiglia 2000; Rintanen 2007; Shin and Davis 2005; Rintanen 2017). A major role in the reduction is played by a representation of *s-t-reachability* of directed graphs as a set of clauses, which is needed for stating the following: for every non-goal state of the execution, if the execution state is reachable from an initial state under the chosen plan, then a goal state is reachable from the state.

The main challenge in this approach is the encoding the s-t-reachability constraint as a propositional formula. These constraints strongly dominate the size of the overall formula, which led Chatterjee et al. to limit the constraints by ad hoc means, without which scalability is poor.

Our first contribution is to show how the size of the encodings can be reduced from quadratic to linear (in the size of the graph representing all possible executions), by replacing the CNF-encoded graph constraints by specialized constraints for graph properties. This change, together with a SAT solver extended with acyclicity constraints, dramatically improves the scalability of the approach, without having to resort to ad hoc limitations of the constraints which compromise the correctness of the approach.

Our second contribution presents two infinite-horizon classes of plans, and establishes tight connections between the different notions of plans and graph constraints: the characterization of plans in terms of properties of graphs representing their behaviors is particularly natural and elegant. The new plan classes with infinite executions extend and strengthen the approach considerably: many applications of AI planning involve unbounded time horizons, for example a robot performing the same task indefinitely. Planning that does not acknowledge that goals are to be reached repeatedly are in these domains insufficient: being able to reach a goal once is not enough, because it is also necessary to guarantee that the goals remain reachable in the future. A simple example is a robot doing cleaning or maintenance in a complex building with limited accessibility of rooms (due to locks in the doors), with the robot performing a task in a way that locks itself in a part of building so that no further tasks can be completed. Other examples are using up or decapacitating critical resources.

To cover such more general scenarios, the concepts used by Chatterjee et al. have to be generalized. We show how this generalization can be very elegantly achieved by using graph constraints. To illustrate the ideas, we first show how the *strong plans* concept – guaranteed one-time reachability

*Also affiliated with Griffith University, Brisbane, Australia, and the Helsinki Institute for Information Technology, Finland. This work was funded by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, 251170).

of goals in a bounded number of steps – of Cimatti et al. (2003) can be expressed in terms of *acyclicity* constraints. Then, we generalize these two finite execution problems to corresponding infinite execution problems, in which goals are repeatedly reached an infinite number of times, respectively with a bounded and an unbounded execution between visits of goal states. Surprisingly, the powerful notions of infinite plans are obtained from the finite execution encodings by very small modifications.

The significance of these results is that earlier works on planning for non-deterministic and partially observable problems limit to finite executions (Bonet and Geffner 2000; Bertoli et al. 2006; Bertoli, Cimatti, and Pistore 2006), with no obvious generalization path to infinite executions by the types of methods used. Hence, we argue that SAT and graph constraints are a conceptually simple and effective approach to these infinite horizon problems, not requiring much on top of the basic framework with finite executions only.

In the experimental part of the work we show that the proposed solution to the quadratic reachability constraints works very well: we observe steep and growing improvements in the runtimes as the instance sizes grow, in accordance with the reduction from quadratic to linear size reachability constraints. The second part of the work is conceptual, and is not experimented with.

The structure of the paper is as follows. In the next section we discuss four different objectives for planning with non-deterministic actions and partial observability. Then we formalize the planning problem in terms of its state space representation and executions under a given plan, followed by definitions of the four objectives that plans may have to satisfy. This is followed by translations of all resulting planning problems into the propositional logic, including the related graph constraints. In the experiments section we investigate the impact of replacing a standard CNF-based SAT solver with one that includes specialized propagators for graph constraints. We conclude by summarizing the results and pointing out topics for future research.

Planning with Partial Observability

Planning with partial observability was perhaps first pursued in the context of partially observable Markov decision processes (POMDP) (Smallwood and Sondik 1973). The high complexity of POMDPs, as well as the lack of numeric probabilities in some applications, has motivated research on partial observability without numeric probabilities (Peot and Smith 1992; Weld, Anderson, and Smith 1998). See the related works section for a brief overview.

In this work we focus on the recent approach by Chatterjee et al. (2016) to planning without numeric probabilities and with partial observability, which is motivated by the need to reduce the overall complexity of planning with partial observability, and which focuses on cases where only a small number of *belief states* are needed, that is, it is possible to reach the goals without having to *remember* much about the past execution of the plan.

This approach has been so far defined for one specific notion of plans, those that reach the specified goal states eventually, meaning that reaching the goals is guaranteed, but there are no finite upper bound on the number of steps that may be needed. This notion of plans was first discussed by Cimatti et al. (2003) – called *strong cyclic* plans by them – in the context of planning with non-deterministic actions and full observability,

Chatterjee et al. (2016) presented an interesting approach to find strong cyclic plans in the partially observable setting, restricted to *small-memory* plans. Their work was based on the observation that solving the class of planning problems they focused on is NP-complete, and therefore solvable by reduction to the propositional satisfiability problem SAT.

In this section we review existing notions of plans which limit to finite executions, and propose two extensions to infinite executions.

Bounded Reachability (Strong Plans)

A non-deterministic planning problem has a *strong plan* if for every state there is an action that guarantees the reduction of the remaining steps to a goal state by at least one. We call this *bounded reachability*, as the plans reach the goals with a bounded number of steps. The concept was originally proposed by Cimatti et al. (2003), for planning with full observability, and they also proposed efficient (low polynomial) time algorithms[1] for finding such plans, starting from the set of goal states (having distance 0 to goal states), and then finding states that are guaranteed to reach goals by one action (states with distance 1), then finding states that are guaranteed to reach either a distance 1 or a distance 0 state by one action, and so on, iteratively covering longer and longer distances.

Eventual Reachability (Strong Cyclic Plans)

Eventual reachability means that goals are eventually reached, with no guarantee of reaching them with any finite number of steps. This concept – called *strong cyclic plans* – was first investigated by Cimatti et al. (2003), again for fully observable problems, to cover cases in which no strong plans exist. Consider the goal of tossing a coin with the goal of the coin landing on heads. There is no certainty that heads will be reached in any finite number of steps. But, assuming that all possible non-deterministic outcomes do actually occur (fairness), the probability of eventually reaching the goal is 1, despite the possibility of an infinite sequence of unsuccessful tosses which has probability 0. To find strong cyclic plans (plans that guarantee eventually reaching the goal states) Cimatti et al. propose a clever iterative algorithm that starts from the set of goal states $S_0 = G$, and iteratively finds larger and larger sets S_i such that for every state $s \in S_i$, there is an action a_s such that taking that action in s always reaches a state in S_i and sometimes reaches a state in S_{i-1}. The iteration ends for an i such that all initial states are included in S_i.

The concept of strong cyclic plans has been generalized to planning with partial observability (Bertoli, Cimatti, and Pistore 2006). Chatterjee et al. (2016) formalized the same

[1] Cimatti et al. propose the use of Binary Decision Diagrams (Bryant 1992) to scale up the approach to very large state spaces, but we do not discuss this possibility in this work.

notion of plans in the partially observable case in terms of s-t-reachability, and showed that with bounded memory the problem is NP-complete and solvable with standard SAT solvers.

Repeated Bounded Reachability

The previous two objectives only cover finite executions. Infinite executions are needed whenever some goal has to be reached repeatedly, with no a priori bound on how many times. The strongest notion of plans in this setting is that of *repeated bounded reachability*, which requires that goal states are reached an infinite number of times, and – similarly to bounded reachability – the number of steps between any two consecutive visits to goal states is finitely bounded.

Repeated Eventual Reachability

Repeated eventual reachability are analogous: goal states are visited repeatedly, but now there is no requirement that the interval between the visits is bounded. It is only required that eventually reaching the goals once more is guaranteed, subject to the fairness assumption, similarly to eventual reachability. Repeated eventual reachability is important when no plans for repeated bounded reachability exist.

Infinite executions are standard in most works on MDPs and POMDPs, where the objective is to collect the maximum expected discounted rewards over an infinite horizon. This objective can be viewed as a natural non-metric counterpart of the plans for MDPs and POMDPs with infinite horizons.

Preliminaries

Definition 1 *A transition system $P = (S, A, O, \delta, Z, I, G)$ consists of*

- *S is a finite set of states,*
- *A is a finite set of actions,*
- *O is a finite set of observations,*
- *$\delta \subseteq S \times A \times S$ is the transition relation,*
- *$Z : S \to O$ is the observation function,*
- *$I \subseteq S$ is the set of initial states, and*
- *$G \subseteq S$ is the set of goal states.*

Memory-Free Plans

A memory-free plan is a function $\pi : O \longrightarrow A$ that, given an observation O of a state $s \in S$, chooses one of the actions. States with the same observation are associated with the same action because $\pi(Z(s)) = \pi(Z(s'))$ whenever $Z(s) = Z(s')$, independently of the past observations and actions.

Bounded-Memory Plans

Sometimes it is necessary to keep track information from the execution history which consists of earlier observations and actions, to be able to choose an appropriate action. This information is characterized by the *memory state*. For example, consider a multi-story building where the agent has to reach the fourth floor. If all floors look identical, the agent needs to count the visited floors to be able to decide, which floor to enter. With no memory (and deterministic plans), the agent would be forced to behave in the same way whenever it encounters the same observation.

Instead of remembering everything in the past, bounded-memory plans allow only a small number of different memory states M. The base case $|M| = 1$ means that there is no memory, that is, alternative past executions cannot be explicitly distinguished from each other at all.

Definition 2 *A bounded-memory plan is a tuple $(M, \pi_\alpha, \pi_\mu, m_0)$ such that the following hold.*

- *M is a finite non-empty set of memory states.*
- *The function $\pi_\alpha : M \times O \longrightarrow A$ is the action selection function which, given a memory state and an observation, chooses an action.*
- *The function $\pi_\mu : M \times O \longrightarrow M$ is the memory update function which, given a memory state and an observation, determines the new memory state.*
- *The initial memory state is $m_0 \in M$.*

The behavior of a transition system under a plan is described by the *execution graph*. It will be later used in defining the different objectives, and also in the encodings of contingent planning in the propositional logic.

Definition 3 (Execution Graph) *The* execution graph *for a transition system P and a plan $(M, \pi_\alpha, \pi_\mu, m_0)$ is a graph (V,E), such that*

- *the set V of nodes consists of all $(s, m) \in S \times M$,*
- *the set $E \subseteq V \times V$ of arcs includes $((s_1, m_1), (s_2, m_2))$ iff $(s_1, a, s_2) \in \delta$ for $\pi_\alpha(m_1, Z(s_1)) = a$ and $\pi_\mu(m_1, Z(s_1)) = m_2$, and*
- *the* initial nodes *of the execution graph are (s, m_0) such that $s \in I$.*

Definition 4 (Bounded Execution Graph) *The* bounded execution graph *is like the execution graph, except that there are no outgoing arcs from goal nodes.*

Formalization of Objectives

We now state the four objectives more formally in terms of the properties of execution graphs. The bounded execution graph is used by the bounded objectives, and the full execution graph is used by the objectives with infinite executions. The differences between the objectives show up in the different graph properties that need to be satisfied.

1. **Eventual Reachability**
   Eventual Reachability requires that if there is a path from an initial node (s_0, m_0) of the bounded execution graph to some (s, m) such that $s \notin G$, then there is a path from (s, m) to a goal node (s_2, m_2) where $s_2 \in G$.
2. **Repeated Eventual Reachability**
   Repeated Eventual Reachability differs from Eventual Reachability simply by looking beyond the goal states: if there is a path from an initial node (s_0, m_0) of the (unbounded) execution graph to some (s, m), then there is a path from (s, m) to some (s_2, m_2), where $s_2 \in G$.
3. **Bounded Reachability**
   Bounded Reachability requires the following.

- The bounded execution graph is acyclic to ensure that the goal is reached in a finite number of steps.
- All non-goal nodes (s, m) (with $s \notin G$) that are reachable from an initial node in the bounded execution graph, have a successor node.

These two conditions guarantee that all executions are finite (because of acyclicity), and because all non-goal nodes have a successor, the last node of every execution must be a goal node.

4. **Repeated Bounded Reachability**
Repeated Bounded Reachability requires the following.
   - All nodes reachable from an initial node (m_0, s_0) have a successor node.
   - Every cycle in the execution graph contains at least one node (s, m) such that $s \in G$. That means, cycles that do not involve a goal state are not allowed.

Testing for the existence of plans under all these objectives and under plans with a fixed memory size is in NP, as demonstrated by the reductions to SAT given later. The decision problems are also NP-hard, which can be proved by simple modifications of earlier proofs (Chatterjee, Chmelik, and Davies 2016). NP-completeness of these problems is what justifies the use of a SAT solver to find plans for them. If the problems were polynomial time (like fully observable MDP problems), using a SAT solver would be an overkill.

Encodings in the Propositional Logic

In this section, we explain how the three new objectives are encoded as a SAT problem. Given $k \geq 1$, we will define a propositional formula which is satisfiable iff there is a plan with memory k.

The encodings rely on implicitly representing all possible plans and all possible execution graphs. The choice of a plan – the choice of actions and memory updates for all observation and memory state combinations – induces one particular execution graph. A plan is determined by a truth-value assignment to variables π_α and π_μ below, and the remaining variables represent the corresponding execution graph (through the variables Arc) and its properties.

Variables used in the Encodings

A number of propositional variables are defined to encode contingent planning problem with bounded memory into SAT. The first two represent the functions π_α and π_μ of bounded-memory plans.

- $\pi_\alpha(m, o, a)$ means that action a is taken in memory state m with observation o.
- $\pi_\mu(m, o, m')$ means that memory state m' follows memory state m with observation o.
- $Arc(s, m, s', m')$ means that there is a transition between state-memory pairs (s, m) and (s', m'), for all $(s, s') \in Y$ where $Y = \{(s, s') \in S \times S | (s, a, s') \in \delta$ for some $a \in A\}$.
- $R^I(s, m)$ means that (s, m) is reachable from an initial node under the current plan.
- $R^G(s, m)$ means that a goal node is reachable from (s, m) under the current plan.

Next we describe the constraints common to all objectives, followed by CNF encodings of reachability constraints, and after that we introduce the constraints specific to each objective.

Constraints Common to All Objectives

The constraints that are common to all objectives guarantee that the arc variables exactly correspond to the execution graph of the plan represented by π_α and π_μ. The constraints in the next sections impose additional constraints specific to the four plan types.

The value assignments for π_α and π_μ have to correspond to the respective functions: the functions have to be defined for states that are reachable, and their values have to be unique.

First we require that both represent functions.

$$\neg(\pi_\alpha(m, o, a_1) \wedge \pi_\alpha(m, o, a_2)) \tag{1}$$

for all $m \in M, o \in O, a_1 \in A, a_2 \in A$ such that $a_1 \neq a_2$.

$$\neg(\pi_\mu(m, o, m_1) \wedge \pi_\mu(m, o, m_2)) \tag{2}$$

for all $m \in M, o \in O, m_1 \in M, m_2 \in M$ with $m_1 \neq m_2$.

For all $(s, m) \in S \times M$, if (s, m) is reachable from an initial node, then at least one of the actions executable in s should be enabled in (s, m) and hence part of the plan.

$$R^I(s, m) \rightarrow \bigvee_{a \in X(s)} \pi_\alpha(m, Z(s), a) \tag{3}$$

Here, $X(s) = \{a \in A | (s, a, s') \in \delta,$ for some $s'\}$.

For every memory state $m \in M$ and observation $o \in O$, at least one of the memory updates is possible.

$$\bigvee_{m' \in M} \pi_\mu(m, o, m') \tag{4}$$

The last constraint states that the execution graph has arc $((s, m), (s', m'))$ if and only if the plan prescribes memory update from m to m' for the observation $Z(s)$ and some action a for state s with a possible transition to s'. It is generated for all $(s, s') \in Y$.

$$Arc(s, m, s', m') \leftrightarrow \left(\pi_\mu(m, Z(s), m') \wedge \bigvee_{a \in X(s,s')} \pi_\alpha(m, Z(s), a) \right) \tag{5}$$

Here, $X(s, s') = \{a \in A | (s, a, s') \in \delta\}$.

Reachability Constraints

The simplest – and approximating – encoding of reachability is the following, expressed for reachability from the initial nodes of the execution graph.

$$R^I(s_0, m_0) \text{ for all initial states } s_0 \tag{6}$$

$$R^I(s,m) \wedge Arc(s,m,s',m') \rightarrow R^I(s',m') \quad \text{for all } (s,s') \in Y, m \in M, m' \in M \tag{7}$$

This is an over-approximation, as $R^I(s,m)$ is true for all nodes (s,m) which are reachable from the initial nodes, but it may also be true for unreachable nodes. However, this encoding is far simpler than the non-approximate encoding we present next, and it is sufficient when we only pose the constraint that some nodes are *unreachable* from the initial states, as is the case with all encodings in this paper. This observation was already used by Chatterjee et al. (2016).

An accurate (non-approximating) encoding of reachability is needed for expressing that from a given node, a goal node is reachable. It is based on calculating the distances to goal nodes.

$$R_0^G(s_g,m) \text{ for all goal states } s_g \text{ and all } m \in M \tag{8}$$

$$\neg R_0^G(s,m) \text{ for all non-goal states } s \text{ and all } m \in M \tag{9}$$

$$R_i^G(s,m) \leftrightarrow \Big(R_{i-1}^G(s,m) \vee \bigvee_{(s',s) \in Y, m' \in M} Arc(s',m',s,m) \wedge R_{i-1}^G(s',m')\Big) \tag{10}$$

Now, a goal node is reachable from (s,m) if and only if $R_N^G(s,m)$ is true, where N is the total number of nodes.[2] We define $R^G(s,m)$ as $R_N^G(s,m)$. The last formulas have a size proportional to the number of arcs in the graph. The index i needs to be instantiated for all $1,\ldots,N$. Hence the size of this encoding is linearly proportional to the product of the number of nodes and the number of arcs.

An alternative to the reachability constraints encoded in CNF is to use ad hoc reachability constraint propagators. GraphSAT (Gebser, Janhunen, and Rintanen 2014b) supports the relevant constraints, and combine them with a standard SAT solver (Glucose/MiniSAT). Extensions of the standard SAT problem are motivated by the poor performance of SAT solvers for problems dominated by graph constraints.

To use these constraints, the input file specifies the set of nodes of a graph and a mapping from a subset of propositional variables to the arcs of the graph, for indicating, which arcs are present. In our case, the nodes are the pairs $(s,m) \in V$, a special initial node n_I, and a special goal node n_G. The propositional variables $Arc(s,m,s',m')$ are mapped to arcs from (s,m) to (s',m'). Additionally, we have arcs from the special initial nodes to the initial nodes of the execution graph, as well as arcs from the goal nodes of the execution graph to the special goal node. The propositional variables for these arcs are always set true by including corresponding unit clauses in the input.

Then some other propositional variables are associated with reachability between given two nodes. In our case, we associate $R^G(s,m)$ with the reachability from (s,m) to the special goal node, and $R^I(s,m)$ with the reachability from the special initial node to (s,m).

This representation works exactly like the CNF encoding given above from the point of view of the propositional variables R^G, R^I and Arc.

Additionally, GraphSAT allows the use of the acyclicity constraint, which requires that the graph formed by the arcs for which the corresponding propositional variable are *true*, is acyclic.

S-T-Reachability by Acyclicity During experimentation with the GraphSAT solver, we noticed that instead of using GraphSAT's specialized s-t-reachability constraints, we could obtain a far better scalability and performance by reducing s-t-reachability to GraphSAT's specialized acyclicity constraints. This seemed to be due to the moderately large size of the graphs and the specifics of our reachability tests. So, instead of the specialized s-t-reachability constraints, we ended up representing reachability in terms of specialized acyclicity constraints through the following reduction.

The reduction is based on the following formulas, the first is for all goal nodes $(s,m) \in G \times M$, and the latter two for all non-goal nodes $(s,m) \in (S \backslash G) \times M$.

$$R^G(s,m) \tag{11}$$

$$R^G(s,m) \leftarrow \bigvee_{(s,s') \in Y, m' \in M} Arc(s,m,s',m') \wedge R^G(s',m') \tag{12}$$

$$R^G(s,m) \rightarrow \bigvee_{(s,s') \in Y, m' \in M) \in V} (Arc(s,m,s',m') \wedge R^G(s',m') \wedge J(s,m,s',m')) \tag{13}$$

In addition, the graph consisting of arcs $((s,m),(s',m'))$ such that the propositional variable $J(s,m,s',m')$ is true is required to be acyclic, expressed by GraphSAT's acyclicity constraint. These variables – the justification variables – express the idea that the justification of reachability cannot be circular: reachability of n_0 and n_1 from some third node cannot be justified simultaneously by arcs from n_0 to n_1 and from n_1 to n_0.

In contrast to the encoding of reachability as (8), (9) and (10), this encoding has a size *linear* in the size of the graph.

(Repeated) Eventual Reachability

The eventual reachability objective is the one handled by Chatterjee et al. (2016). For this objective the executions are finite, and we use the bounded execution graph without outgoing arcs from goal nodes. For the repeated eventual reachability objective the (unbounded) execution graph is used instead, with no other differences.

The only additional constraint that is needed on top of the constraints shared by all objectives says that if $(s,m) \in (S \backslash G \times M)$ is reachable from the initial node, then a goal node is reachable from (s,m).

$$R^I(s,m) \rightarrow R_N^G(s,m) \tag{14}$$

[2] A lower value than N could substantially reduce the formula size, but choosing such a lower value is difficult in general because it would be at least as high as the longest simple path in the execution graph of any valid plan.

Bounded Reachability

For the bounded reachability objective, we use the bounded execution graph, and for the boundedness condition to be satisfied, there may not be infinite loops. This is guaranteed by requiring that the execution graph is acyclic, directly achieved with the acyclicity constraint. Since formula (3) guarantees that all reachable states have a successor state, and there is by formula (5) correspondingly a successor state (except for the goal states because we are using the bounded execution graph), all of the executions of a plan must end in a goal state, and all executions have a length bounded by the number of nodes in the execution graph.

Repeated Bounded Reachability

The requirement here is to reach the goal states repeatedly, necessarily inducing cycles in the execution graph, but no cycles without the goal states are allowed. This is achieved simply by imposing the acyclicity constraint on those arcs in the execution graph that do not start from a goal node.

Sizes of Encodings

Asymptotic size upper bounds of the encodings follow.

category	formula	asymptotic size
base encoding	(1)	$\|M\| \times \|O\| \times \|A\|^2$
	(2)	$\|M\|^3 \times \|O\|$
	(3)	$\|S\| \times \|M\| \times \|A\|$
	(4)	$\|M\|^2 \times \|O\|$
	(5)	$\|Y\| \times \|M\|^2 \times \|A\|$
CNF non-s-t-reachability	(6)	$\|S\|$
	(7)	$\|M\|^2 \times \|Y\|$
CNF s-t-reachability	(8)	$\|S\| \times \|M\|$
	(9)	$\|S\| \times \|M\|$
	(10)	$\|S\| \times \|M\|^2 \times \|Y\|$
reachability by acyclicity	(11)	$\|S\| \times \|M\|$
	(12)	$(\|S\| \times \|M\|) + \|Y\|$
	(13)	$(\|S\| \times \|M\|) + \|Y\|$
(repeated) eventual reachability	(14)	$\|S\| \times \|M\|$

The formulas (1) and (2) can be respectively reduced by factors $|A|$ and $|M|$ by using standard cardinality constraint encodings (Sinz 2005), which is also what our implementation does. Assuming that M and O are small, the base encoding is effectively linear in the size of the graph that depicts all transitions between states, labelled with the possible actions (this does not fully become visible in the size characterization of (5).)

The improvement from representing the reachability constraint in CNF (formulas (8), (9) and (10)) to representing it in terms of acyclicity (formulas (11), (12) and (13)) is substantial, essentially from quadratic to linear in the size of the state-space graph underlying the planning problem, as also shown in Figure 2.

Chatterjee et al. (2016) tried to solve the problem of very large reachability encodings by manually determining – through trial and error – bounds N for the formulas (10) that are far lower than the longest simple paths in the graph. No theoretical or practical justification was given for this. For example, for the satisfiable Escape instances they considered a state to be unreachable if there are no paths of length $\leq N = 2$, which is certainly far lower than the maximum path lengths in the belief space. The Escape problems have belief space sizes up to several thousands, and the longest paths without repeating states in the state space have lengths in the dozens, which would be a lower bound on acceptable N. In contrast, we used $N = |S \times M|$, the number all states of execution. This leads to higher runtimes and larger formulas, but is more realistic because tight upper bounds on the required N parameter are in general not available.

Experiments

The goal of our experiments is to demonstrate that the poor scalability of quadratic encodings of reachability constraints, as used by Chatterjee et al. (2016), is avoided by using specialized constraint propagators in some SAT solvers.

With the quadratic size of the constraints, for problems with thousands of states, formula sizes quickly climb to hundreds of megabytes and further, making their generation and use slow. Second, the complexity and size of the constraints are an obstacle of solving the planning problems efficiently. Our experiments show that both problems are eliminated: the formulas with the linear encodings are dramatically smaller, and most of the time solved in a fraction of a time in comparison to the original formulas.

We implemented a program to translate contingent planning problems into the language used by GraphSAT, which is the DIMACS CNF format extended with graph constraints. We also implemented a translation from GraphSAT's input language into the standard DIMACS CNF format, to be able to compare specialized graph constraint propagators to the existing CNF encodings of the same constraints. Given the reachability constraints in the GraphSAT syntax, this translator produces exactly the Chatterjee et al. reachability encoding, including the compact non-reachability encoding without node distances.

We hypothesized that solving the planning problems with specialized graph constraint propagators would outperform the purely clausal encodings used by Chatterjee et al. (2016). This hypothesis turned out to be true. However, very surprisingly, reducing reachability to acyclicity – with the reduction given earlier – dramatically outperformed the specialized reachability constraints. For this reason, in the following we will not be reporting on the specialized reachability constraints of GraphSAT.

We devised a collection of grid navigation problems similar to the ones used by Chatterjee et al. (2016), which in turn are variants of problems proposed earlier (McCallum 1993). In our problems a robot is moving in a grid north, south, east or west, and attempts to reach a target grid cell, when the initial location is unknown and randomly chosen. The robot can observe the wall in immediately neighboring cells in the four cardinal directions. These are our problem classes *comb*, *emptycorner*, *emptymiddle* and *roomchain*, with respectively a comb-shaped grid map, empty grid with goal in the corner (with $p \times p$ grid cells with parameter p as indicated in the instance name), empty grid with goal in the middle (same p parameter), and a sequence of rooms with goal in the middle room. For the comb-shaped mazes the planner discovers the "follow the wall on the left/right" plan, and for

π_α	E	NE	NW	N	SE	SW	S	W	-
0	N	S	S	S	N	E	N	S	S
1	W	N	S	W	S	W	E	S	N
π_μ	E	NE	NW	N	SE	SW	S	W	-
0	0	1	0	0	0	1	1	1	0
1	1	1	0	0	0	0	1	1	1

Table 1: Plan with 2 memory states for scanning a grid

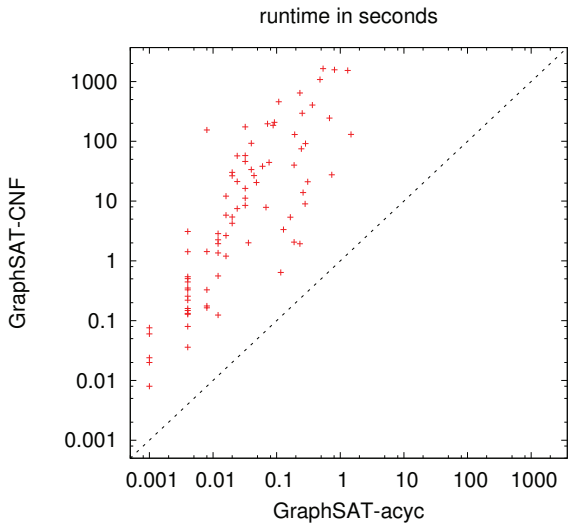

Figure 1: Runtimes for CNF constraints vs. encoding with acyclicity

the others it discovers various policies for example scanning the grid systematically vertically or horizontally, or going to the corner and then diagonally zigzagging to the goal cell if it is in the middle. We tried grids of different sizes, up to some hundreds of grid cells. A sample plan for scanning a rectangular grid for *emptymiddle* is depicted in Table 1.

Additionally, we formalized a pursuit-evasion scenario in an empty rectangular grid (similar to one used by Chatterjee et al. (2016)), which we called *newescape*, in which the robot has to avoid being hit by another robot that moves unpredictably. The robot can observe the presence of the pursuing robot in neighboring grid cells. The robot survives by staying in place until the pursuing robot comes next to it, after which it needs to move away.

All our experiments are with the *repeated eventual reachability* objective, as the pursuit-evasion problem is interesting only with infinite executions. The other problems have solutions also for the finite objectives.

All runs were with a 3600 second time limit and performed on Intel Xeon E3-1230 V2 CPUs at 3.30 GHz.

We tested an increasing number of memory states until the instances became solvable (respectively 3, 1, 2, 2 or 3, 1 memory states for the 5 problems, independent of the grid size except for the *roomchain* problem.) The graphs have a size up to some thousands of nodes. The runtimes of the

instance	value	CNF	Acyc
comb006-1	F	0.02	0.00
comb006-2	F	3.34	0.13
comb006-3	T	9.00	0.28
comb008-1	F	0.08	0.00
comb008-2	F	39.95	0.19
comb008-3	T	1573.07	0.81
comb010-1	F	0.16	0.00
comb010-2	F	129.81	0.19
comb010-3	T	1533.21	1.31
emptycorner020-1	T	12.11	0.02
emptycorner030-1	T	26.70	0.04
emptycorner040-1	T	458.54	0.11
emptycorner050-1	T	294.36	0.25
emptymiddle005-1	F	0.04	0.00
emptymiddle005-2	T	0.12	0.01
emptymiddle010-1	F	1.42	0.00
emptymiddle010-2	T	44.38	0.08
emptymiddle015-1	F	57.14	0.02
emptymiddle015-2	T	1641.48	0.54
emptymiddle020-1	F	174.56	0.03
emptymiddle020-2	T	-	0.39
roomchain002-1	F	0.13	0.00
roomchain002-2	T	0.64	0.12
roomchain003-1	F	0.26	0.00
roomchain003-2	F	205.13	-
roomchain003-3	T	-	26.18
roomchain004-1	F	0.56	0.01
roomchain004-2	F	-	-
roomchain004-3	T	-	118.86
newescape02-1	T	0.01	0.00
newescape03-1	T	0.16	0.01
newescape04-1	T	1.20	0.02
newescape05-1	T	7.85	0.07
newescape06-1	T	38.17	0.06
newescape07-1	T	19.89	0.23
newescape08-1	T	58.55	0.37
newescape09-1	T	159.20	0.46

Table 2: Runtimes with GraphSAT

experiments are reported in Table 2, with the same data depicted in Figure 1 to visualize the general trend, and Figure 2 illustrating the formula size reductions.

As can be seen from Figure 1, representing the reachability constraints in terms of acyclicity constraints dramatically reduces the runtimes in comparison to the CNF-encoded reachability constraints, often by 2 or 3 orders of magnitude. GraphSAT with acyclicity constraints scales far further than the instances shown in Table 2 and what we experimented with, in all categories except the *roomchain* problem. Roomchain is also the only problem in which CNF-encoded reachability sometimes outperforms the reduction to acyclicity (the instance *roomchain03-2*, where only the CNF encoding can determine that > 2 memory states are needed.)

For all categories, the sizes of largest instances with CNF-encoded reachability are in the gigabyte range. However, in most cases the runtimes are a more limiting factor than the formula size. An exception is the *newescape* problem, in which newescape06-1 has a size of 3.6 GB although it is rel-

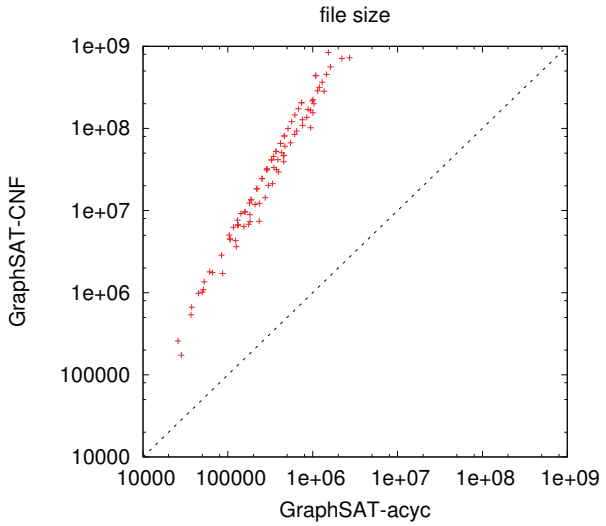

Figure 2: File sizes for CNF constraints vs. encoding with acyclicity

atively quickly solvable. Larger instances in this series soon become impractically large. See Figure 2 for a comparison of file sizes with CNF-encoded reachability constraints and reachability reduced to acyclicity. The size difference grows as the instance size grows, and reaches three orders of magnitude for the largest instances, that is, 1000 times larger formulas for CNF-encoded reachability constraints.

Related Work

Planning with partial observability has been early on investigated in the framework of Partially Observable Markov Decision Processes (POMDP) (Smallwood and Sondik 1973). Due to the high complexity of this framework (Mundhenk et al. 2000; Madani, Hanks, and Condon 2003), various abstractions of the problem have been proposed, including ignoring the probabilities. Without probabilities, beliefs are sets of states, and complexity is reduced (Rintanen 2004). A basic approach in this vein is AND-OR tree search with belief states (Bonet and Geffner 2000; Bertoli et al. 2006; To, Pontelli, and Son 2011). Belief space search has also been considered in backward direction (Weld, Anderson, and Smith 1998; Rintanen 2005). Most works consider the Bounded Reachability objective only: reaching the goals has to be guaranteed within a bounded number of actions.

In connection with partial observability, eventual reachability or strong cyclic plans have been considered by Bertoli et al. (Bertoli, Cimatti, and Pistore 2006).

The idea of bounded memory plans has earlier been extensively pursued in the context of POMDPs (Lusena et al. 1999; Meuleau et al. 1999). Many practically important problems only require a small amount of memory, justifying the approach. In addition to reduction in complexity, small memory plans can be useful because they are smaller and easier to understand.

Other means to limit the complexity caused by belief states have been proposed. Albore et al. (2009) show that bounds on the *contingent width* make efficient reduction to planning with full observability possible, eliminating the need to handle belief states explicitly.

Graph constraints occur in many applications. Acyclicity is needed for example in representations of graphical models for machine learning (Cussens 2008; Corander et al. 2013), partial-order methods for SAT-based reachability (Rintanen, Heljanko, and Niemelä 2006), and in knowledge representation languages with non-monotonic negation (Gebser, Janhunen, and Rintanen 2014a).

Conclusion

We have investigated the close connection between graph constraints and contingent planning with small-memory plans, first showing how generalized notions of plans, including infinite executions, can be covered by using natural graph-theoretic concepts, and then shown through experimentation that specialized solvers for graph constraints improve the scalability of the approach dramatically, in many cases almost trivializing the constraint solving process when the basis approach can barely find solutions.

Small-memory plans, whenever applicable, are interesting both for their potential to reduce complexity and their small size in comparison to unrestricted conditional plans.

For future work we have started to pursue the application of Mixed Integer Linear Programming (MILP) and SMT solvers to solving small-memory POMDPs. Expressing transition probabilities and real-valued rewards is straightforward in these frameworks, and together with optimization it is possible to find the highest-reward POMDP solutions for a given memory size.

References

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 1623–1628. AAAI Press.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2006. Strong planning under partial observability. *Artificial Intelligence* 170(4):337–384.

Bertoli, P.; Cimatti, A.; and Pistore, M. 2006. Strong cyclic planning under partial observability. In *ECAI 2006 – 17th European Conference on Artificial Intelligence*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, 580. IOS Press.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, 52–61. AAAI Press.

Bryant, R. E. 1992. Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys* 24(3):293–318.

Chatterjee, K.; Chmelik, M.; and Davies, J. 2016. A symbolic SAT-based algorithm for almost-sure reachability with small strategies in POMDPs. In *Proceedings of the*

30th AAAI Conference on Artificial Intelligence (AAAI-16), 3225–3232. AAAI Press.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(1–2):35–84.

Corander, J.; Janhunen, T.; Rintanen, J.; Nyman, H.; and Pensar, J. 2013. Learning chordal Markov networks by constraint satisfaction. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems 26*, 1349–1357.

Cussens, J. 2008. Bayesian network learning by compiling to weighted MAX-SAT. In *UAI'08, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, 105–112.

Ferraris, P., and Giunchiglia, E. 2000. Planning as satisfiability in nondeterministic domains. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000) and the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-2000)*, 748–753. AAAI Press.

Gebser, M.; Janhunen, T.; and Rintanen, J. 2014a. Answer set programming by SAT modulo acyclicity. In *ECAI 2014. Proceedings of the 21st European Conference on Artificial Intelligence*, 351–356. IOS Press.

Gebser, M.; Janhunen, T.; and Rintanen, J. 2014b. SAT modulo graphs: Acyclicity. In Fermé, E., and Leite, J., eds., *Logics in Artificial Intelligence, 14th European Conference, JELIA 2014, September 2014, Proceedings*, volume 8761 of *Lecture Notes in Computer Science*, 137–151. Springer-Verlag.

Kautz, H., and Selman, B. 1996. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, 1194–1201. AAAI Press.

Lusena, C.; Li, T.; Sittinger, S.; Wells, C.; and Goldsmith, J. 1999. My brain is full: When more memory helps. In *Uncertainty in Artificial Intelligence, Proceedings of the Fifteenth Conference (UAI-99)*, 374–381. Morgan Kaufmann Publishers.

Madani, O.; Hanks, S.; and Condon, A. 2003. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence* 147(1–2):5–34.

McCallum, R. A. 1993. Overcoming incomplete perception with utile distinction memory. In *Machine Learning: Proceedings of the Tenth International Conference on Machine Learning*, 190–196. Morgan Kaufmann Publishers.

Meuleau, N.; Kim, K.-E.; Kaelbling, L. P.; and Cassandra, A. R. 1999. Solving POMDPs by searching the space of finite policies. In *Uncertainty in Artificial Intelligence, Proceedings of the Fifteenth Conference (UAI-99)*, 417–426. Morgan Kaufmann Publishers.

Mundhenk, M.; Goldsmith, J.; Lusena, C.; and Allender, E. 2000. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM* 47(4):681–720.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3).

Peot, M. A., and Smith, D. E. 1992. Conditional nonlinear planning. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, 189–197. Morgan Kaufmann Publishers.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence* 170(12-13):1031–1080.

Rintanen, J. 2004. Complexity of planning with partial observability. In *ICAPS 2004. Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, 345–354. AAAI Press.

Rintanen, J. 2005. Conditional planning in the discrete belief space. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 1260–1265. Morgan Kaufmann Publishers.

Rintanen, J. 2007. Asymptotically optimal encodings of conformant planning in QBF. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI-07)*, 1045–1050. AAAI Press.

Rintanen, J. 2017. Temporal planning with clock-based SMT encodings. In *IJCAI 2017, Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 743–749. AAAI Press.

Shin, J.-A., and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artificial Intelligence* 166(1):194–253.

Sinz, C. 2005. Towards an optimal CNF encoding of Boolean cardinality constraints. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming, Sitges, Spain, October 2005*. Springer-Verlag. 827–831.

Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21:1071–1088.

To, S. T.; Pontelli, E.; and Son, T. C. 2011. On the effectiveness of CNF and DNF representations in contingent planning. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2033–2038. AAAI Press.

Weld, D. S.; Anderson, C. R.; and Smith, D. E. 1998. Extending Graphplan to handle uncertainty and sensing actions. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, 897–904. AAAI Press.