

Belief Tracking for Planning with Sensing

Blai Bonet

Universidad Simón Bolívar

2nd Brazilian Conf. on Intelligent Systems (BRACIS)

Fortaleza, Brazil 2013

(joint work with Hector Geffner)



Recap on Early Days of AI: Programming and Methodology

Many of the contributions had to do with:

- programming
- representation and use of knowledge in programs

It was common to find dissertations in AI that:

- pick up a task and domain X
- analyze how the task is solved
- capture this reasoning in a **program**

The dissertation was

- a **theory** about X , and
- a **program** implementing the theory, **tested** on a few examples

Great ideas came out ... but there was a problem ...

Methodological Problem: Generality

Theories expressed as programs are **not falsifiable**:

- ▶ when program fails, the blame is on **'missing knowledge'**

Three approaches to this problem:

- narrow the domain (expert systems)

- ▶ **problem:** lack of generality

- accept the program as an illustration, a demo

- ▶ **problem:** limited scientific value

- fill up the missing value (using intuition, commonsense, ...)

- ▶ **problem:** not clear how to do; not successful so far

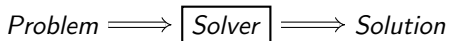
AI Research Today

Recent works in AIJ, JAIR, AAI or IJCAI are on:

- **SAT and Constraints**
- **Search and Planning**
- **Probabilistic Reasoning**
- **Probabilistic Planning**
- Multi-Agent Systems
- Inference in First-Order Logic
- Machine Learning
- Natural Language
- Vision and Robotics
- ...

First four areas often deemed as **techniques**, but it is more accurate to think about them in terms of **models and solvers**

Example: Solver for Linear Equations



Problem: the age of John is 3 times the age of Peter. In 10 years, it will be only 2 times. How old are John and Peter?

Expressed as: $J = 3P$; $J + 10 = 2(P + 10)$

Solver: Gauss-Jordan (Variable Elimination)

Solution: $P = 10$; $J = 30$

Solver is **general** as deals with any instance of the **model** (linear equations)

The linear equations model is **tractable**; AI models are not . . .

Example from AI: Solvers for SAT

CNF instance \implies SAT Solver \implies *Solution*

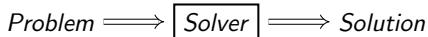
SAT is the problem of determining whether there is a **truth assignment** that satisfies a set of clauses

$$x \vee y \vee \neg z \vee \neg w \vee \dots$$

Problem is **NP-Complete**: this means worst-case behavior of SAT algorithms is **exponential** in number of variables ($2^{100} = 10^{30}$)

Current SAT solvers tackle problems with **thousands of variables and clauses**, and are used widely (circuit design, verification, planning, etc)

AI Models and Solvers



Some basic models and solvers currently considered in AI:

- **CSP/SAT**: find state that satisfies constraints
 - **Bayesian Networks**: find probability over variable given observations
 - **Planning**: find action sequence or policy that produces desired state
-
- ▶ Solvers for these models are general; not **tailored** to specific instances
 - ▶ Models are all **intractable**
 - ▶ Solvers all have a **clear and crisp** scope: instances of the model
 - ▶ Challenge is mainly **computational**: how to scale up
 - ▶ Methodology is **empirical**: benchmarks and competitions

How SAT solvers do it?

Two types of **efficient (polytime) inference** at every node of search tree:

- unit resolution
- conflict-based clause learning and backtracking

Other ideas are possible but **don't work** (i.e. don't scale up):

- generate and test each possible assignments (**pure search**)
- apply general resolution (**pure inference**)

Basic Planning Model and Task

Planning is the **model-based approach** to autonomous behavior:

- a system can be in one of many **states**
- states assign **values** to a set of **variables**
- **actions** change the values of certain variables

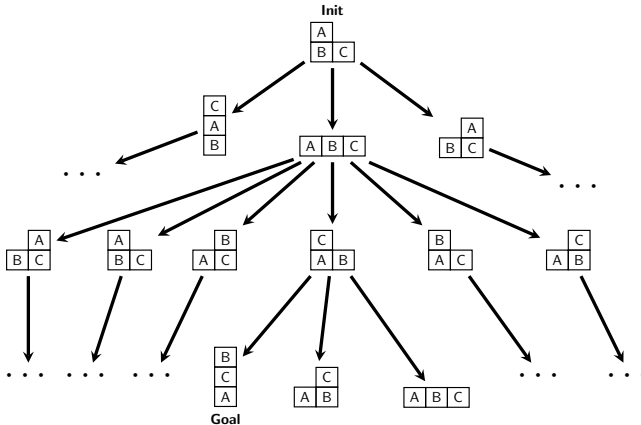
Basic task: find **action sequence** to drive **initial state** into **goal state**

Model instance \implies *Planner* \implies *Action sequence*

Complexity: NP-hard; i.e., exponential in number of vars in **worst case**

Box is generic: should work on any instance no matter what it is about

Example: Blockworld



Task: given actions that move a 'clear' block to the table or onto another 'clear' block, **find a plan** to achieve given goal

Question: how to find a path in graph of **exponential size** in # blocks?

Summary

- ▶ Research agenda is clear: develop **solvers** for a class of **models**
- ▶ **Solvers** unlike other programs are **general**: they don't target individual problems but families of problems (**models**)
- ▶ Main challenge is **computational**: how to scale up
- ▶ **Structure** of problems must be recognized and **exploited**
- ▶ Progress is measured **empirically**

Agenda for the Rest of the Talk

- ▶ Introduction to planning models and languages
- ▶ Planning under uncertainty: non-det actions and incomplete information
- ▶ Belief tracking in planning
- ▶ Discussion

Planning Models and Languages

Autonomous Behavior in AI

The key problem is to select the **action to execute next**. This is the so-called **control problem**.

Three approaches to the control problem:

- **Programming-based:** specify control by hand
 - ▶ **Advantage:** domain-knowledge easy to express
 - ▶ **Disadvantage:** cannot deal with situations not anticipated by programmer
- **Learning-based:** learn control from experience
 - ▶ **Advantage:** does not require much knowledge in principle
 - ▶ **Disadvantage:** in practice, right features needed, incomplete information is problematic, and unsupervised learning is slow
- **Model-based:** specify problem by hand, derive control automatically
 - ▶ **Advantage:** flexible, clear, and domain-independent
 - ▶ **Disadvantage:** **need a model**; computationally **intractable**

Model-based approach to intelligent behavior called Planning in AI

Classical Planning: Simplest Model

- finite **state space** S
- **known** initial state $s_0 \in S$
- subset $S_G \subseteq S$ of **goal states**
- actions $A(s) \subseteq A$ executable at state s
- **deterministic** transition function $f : S \times A \rightarrow S$ such that $f(s, a)$ is state after applying action $a \in A(s)$ in state s
- non-negative costs $c(s, a)$ for applying action a in state s

Solution is **sequence of actions** (path) that map initial state into goal

Its cost is the sum of costs of the actions in the sequence

Abstract model that works at 'flat' representation of problem

Probabilistic Planning: Markov Decision Processes (MDPs)

- finite state space S
- known initial state $s_0 \in S$
- subset $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ executable at state s
- **transition probabilities** $P(s'|s, a)$ of reaching state s' after applying action a in state s
- non-negative costs $c(s, a)$ for applying action a in state s

Solution can't be linear; it is **function (policy)** that maps states to actions

Cost of solution is **expected cost** to reach goal from initial state

Partially Observable MDPs (POMDPs)

POMDPs are probabilistic models that are **partially observable**

- finite state space S
- initial **distribution (belief)** b_0 over states
- subset $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ executable at state s
- transition probabilities $P(s'|s, a)$ for each states s, s' and action $a \in A(s)$
- finite set of **observable tokens** O
- **sensor model** given by probabilities $P(o|s', a)$ for observing token $o \in O$ after reaching s' when last action done is a

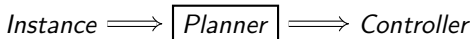
Solution is **policy** mapping belief states (distributions) into actions

Cost of solution is **expected cost** to reach goal from initial distribution

Planners

A planner is a **solver** over a **class of models**

- input is a model description
- output is a controller (solution)



Different models and solution forms: uncertainty, feedback, costs, ...

Instance described with **planning language** (Strips, PDDL, PPDDL, ...)

Factored Languages

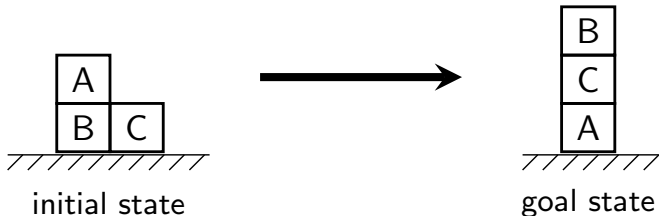
Model specified in **compact form** using high-level language

Language based on propositional variables:

- finite set F of propositional variables (atoms)
- an initial state $I \subseteq F$
- a goal description $G \subseteq F$
- finite set A of operators; each operator $a \in A$ given by
 - ▶ **precondition** that tell action applicable at each state
 - ▶ **effects** that define transition function (i.e. $f(s, a)$ or $F(s, a)$)
- non-negative costs $c(a)$ for applying actions $a \in A$

Language based on **multi-valued variables**: instead of boolean variables, uses variables X with finite domain D_X

Example: Blockworld



Atoms: Clear(?x), On(?x,?y), OnTable(?x)

Actions: Move(?x,?y,?z), MoveToTable(?x), MoveFromTable(?x,?y)

Example: Blocksworld in PDDL

```
(define (domain BLOCKS)
  (:requirements :strips)
  (:predicates (clear ?x) (on ?x ?y) (ontable ?x))

  (:action move
    :parameters (?x ?y ?z)
    :precondition (and (clear ?x) (clear ?z) (on ?x ?y))
    :effect (and (not (clear ?z)) (not (on ?x ?y)) (on ?x ?z) (clear ?y)))

  (:action move_to_table
    :parameters (?x ?y)
    :precondition (and (clear ?x) (on ?x ?y))
    :effect (and (not (on ?x ?y)) (clear ?y) (ontable ?x)))

  (:action move_from_table
    :parameters (?x ?y)
    :precondition (and (ontable ?x) (clear ?x) (clear ?y))
    :effect (and (not (ontable ?x)) (not (clear ?y)) (on ?x ?y)))
)

(define (problem BLOCKS_3_1)
  (:domain BLOCKS)
  (:objects A B C)
  (:init (clear A) (clear C) (on A B) (ontable B) (ontable C))
  (:goal (and (on B C) (on C A))))
```

From Language to Model

Problem $P = \langle F, A, I, G, c \rangle$ mapped into model $\mathcal{S}(P) = \langle S, A, f, s_0, S_G, c \rangle$:

- states S are all the 2^n **truth-assignments** to atoms in F , $|F| = n$
- initial state s_0 assigns **true** to all $p \in I$ and **false** to all $p \notin I$
- goal states are assignments satisfying the goals in G
- executable actions at state s are $A(s) = \{a : s \models pre(a)\}$
- outcome $f(s, a)$ defined by action's effects (in standard way)
- costs $c(a)$

Size of state model is **exponential** in size of problem P (e.g. blocksworld)

State of the Art in Classical Planning

Solution is path from initial state to goal in an **exponential graph**

State-of-the-art algorithms do **search in implicit graph** using heuristics to guide the search

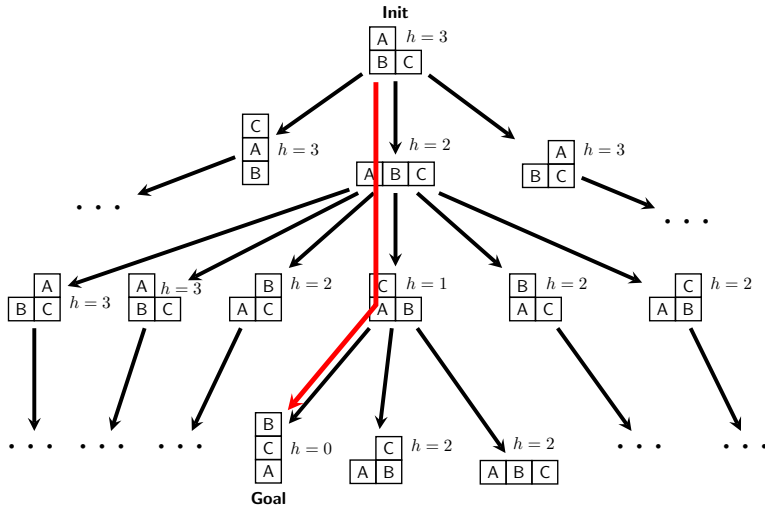
Powerful **heuristics automatically extracted** from problem description

Approach is **general and successful**: able to solve large problems quickly

Planners: LAMA-11, FF, ... (publicly available)

Benchmarks: thousands ... IPC repository (over 80 domains / 3,500 problems)

Finding Solutions: Blocksworld



Planning under Uncertainty

Motivation

Classical planning works: able to solve very large problems

Model is simple, but useful:

- ▶ operators may be non-primitive; abstractions of policies
- ▶ **closed-loop replanning** is able to cope with uncertainty sometimes

There are some limitations, though:

- ▶ can't model **uncertainty on outcome of actions**
- ▶ can't deal with **incomplete information** (partial sensing)
- ▶ ...

Two ways of handling limitations:

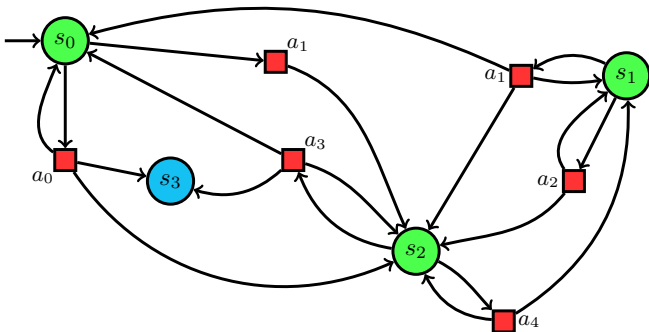
- ▶ **extend scope** of current classical solvers (translations / compilation)
- ▶ develop **new solvers for extended models**

(Fully Observable) State Model with Non-Det Actions

- finite state space S
- known initial state s_0
- goal states $S_G \subseteq S$
- actions $A(s) \subseteq A$ executable at state s
- **non-deterministic** transition function $F : S \times A \rightarrow 2^S$ such that $F(s, a)$ is subset of states that **may** result after executing a at s
- non-negative costs $c(s, a)$ of applying action a in state s

Current state is always fully observable to agent

Example: Simple Problem (AND/OR Graph)



- 4 states: $S = \{s_0, \dots, s_3\}$

- 5 actions: $A = \{a_0, a_1, a_2, a_3, a_4\}$

- 1 goal: $S_G = \{s_3\}$

- $A(s_0) = \{a_0, a_1\}$; $A(s_1) = \{a_1, a_2\}$

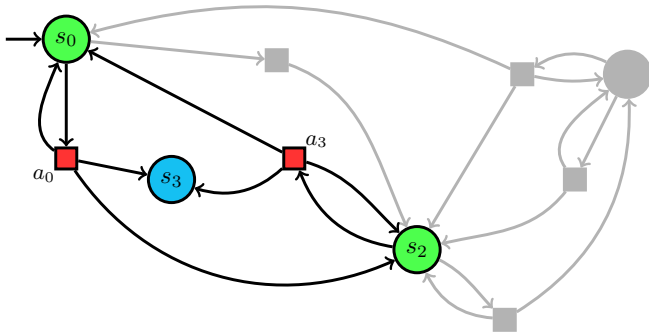
- $F(s_0, a_0) = \{s_0, s_2, s_3\}$

- $F(s_1, a_1) = \{s_0, s_1, s_2\}$

- $F(s_0, a_1) = \{s_2\}$

- ...

Example: Solution



Controller π :

- initial state s_0
- $\pi(s_0) = a_0$
- $\pi(s_2) = a_3$

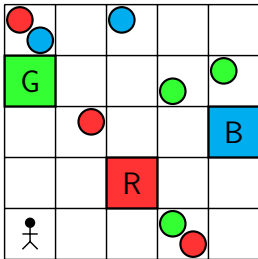
Agent with Partial Information

Agent has **partial information** when it doesn't **fully see current state**

Different ways to model sensing; most frequent based on POMDP model:

- finite set O of **observable tokens**
- **environment produces** one such token **after action is applied**
- agent **receives token** (it doesn't see state directly)
- token may depend on **current state** and **action leading to it**

Example: Collecting Colored Balls



Agent senses presence of balls (and their colors) in current cell

Observable tokens $O = \{000, 001, 010, \dots, 111\}$ (i.e. 3 bits of information)

- **First bit** tells whether there is a **red ball** in same cell of agent
- **Second bit** tells whether there is a **green ball** in same cell of agent
- **Third bit** tells whether there is a **blue ball** in same cell of agent

Model for Non-Det Planning with Sensing (Logical POMDPs)

- finite state space S
- subset of possible initial states $S_I \subseteq S$
- subset of goal states $S_G \subseteq S$
- actions $A(s) \subseteq A$ executable at state s
- non-deterministic transition function $F : S \times A \rightarrow 2^S$
- finite set of **observable tokens** O
- **sensor model** $O(s', a) \subseteq O$ where $O(s', a)$ is non-empty set of possible tokens after a leading to state s : i.e.

transition $s \xrightarrow{a} s'$ generates observable token from $O(s', a)$

- non-negative costs $c(s, a)$ for applying action a in state s

Belief States and Belief Tracking

Agent must keep track of **possible current states** in the form of a **subset of states**; such subsets are called **belief states**

The initial belief state is $b_0 = S_I$ (possible initial states)

When agent has belief state b , then

– **after executing action a ,**

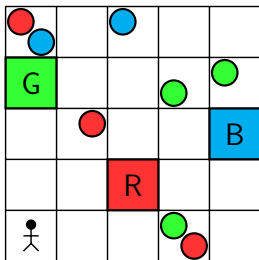
$$b_a = \{s' : s' \in F(s, a) \text{ and } s \in b\} \quad (\text{progression})$$

– **after executing action a and receiving token o ,**

$$b_a^o = \{s' \in b_a : o \in O(s', a)\} \quad (\text{filtering})$$

Beliefs states depend on history of actions and observations!

Example: Belief Tracking on Collecting Colored Balls



► Initial belief $b_0 = \{\text{states w/ agent at } (0,0) \text{ and no balls at } (0,0)\}$ $|b_0| \approx 10^{10}$

► For belief $b = b_0$ and action $a = up$,

$b_a = \{\text{states w/ agent at } (0,1) \text{ and no balls at } (0,0)\}$ $|b_a| \approx 10^{10}$

► Then, agent receives the observation $o = 100$,

$b_a^o = \{\text{states w/ agent at } (0,1), \text{ no balls at } (0,0), \text{ and red balls at } (0,1)\}$ $|b_a^o| \approx 10^9$

POMDPs as Non-Deterministic Planning in Belief Space

From model $P = \langle S, A, F, S_I, S_G, O, c \rangle$, construct **fully observable non-deterministic** model in **belief space** $\mathcal{B}(P) = \langle S', A', F', s'_0, S'_G, c' \rangle$

- states S' are all the belief states
- initial state s'_0 is initial belief
- goal states S'_G are beliefs that only deem possible goals in S_G
- actions $A'(b) = \{a : a \in A(s) \text{ for states } s \text{ deemed possible by } b\}$
- non-deterministic transitions $F'(b, a) = \{b'_a : o \text{ is possible after } a \text{ in } b\}$
- action costs $c'(b, a) = \max_{s \in b} c(s, a)$

Akin to determinization of Non-det. Finite Automata (NFA)

Language for Planning with Sensing

- V is finite set of variables X , each with finite domain D_X
- initial states given by **clauses** I
- goal description G that is **partial valuation**
- finite set A of actions with prec. and non-deterministic cond. effects
- **observable variables** V' (not necessarily disjoint from V)
- **sensing formulas** $W_a(Y = y)$ for each action a and literal $Y = y$
- non-negative costs $c(a)$ for applying action a

Observable tokens are full valuations over observable variables V'

Algorithms: Finding Solutions

Algorithms perform some type of **search** in either

- state space
- belief space

	deterministic	non-deterministic
full obs.	state space / OR graph	state space / AND/OR graph
no obs.	belief space / OR graph	belief space / OR graph
partial obs.	belief space / AND/OR graph	belief space / AND/OR graph

Belief Tracking

Motivation

Want to develop solvers for problems with non-det. and partial sensing

Two **fundamental tasks** to be solved (both intractable):

- tracking of belief states (i.e. representation of search space)
- action selection for achieving the goal (i.e. type of search)

[B & Geffner, AAAI 2012; B & Geffner, IJCAI 2013]

Belief Tracking Pops Up Everywhere

- Simultaneous Localization and Mapping (SLAM) in robotics
- Adversarial games ; Partially Observable Stochastic Games (POSGs)
- Context-based disambiguation in NLP: Hidden Markov Models (HMMs)
- Target tracking and control theory: Kalman filter
- Activity recognition
- Gene prediction, protein folding
- ...

Belief Tracking in Planning (BTP)

Definition (BTP)

Given execution $\tau = \langle a_0, o_0, a_1, o_1, \dots, a_n, o_n \rangle$ **determine** whether

- the execution τ is possible, and
- whether b_τ , the belief that results of executing τ , achieves the goal

Theorem

BTP is NP-hard and coNP-hard

Basic Algorithm: Flat Belief Tracking

Explicit representation of beliefs states as sets of states

Definition (Flat Belief Tracking)

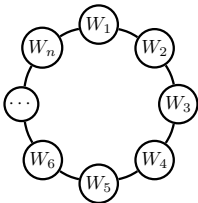
Given belief b at time t , and action a (applied) and observation o (obtained), the belief at time $t + 1$ is the belief b_a^o given by:

$$b_a = \{s' : s' \in F(s, a) \text{ and } s \in b\}$$

$$b_a^o = \{s' : s' \in b_a \text{ and } s' \models W_a(\ell) \text{ for each } \ell \text{ s.t. } o \models \ell\}$$

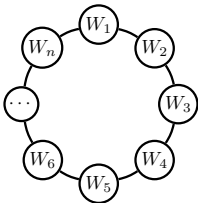
- ▶ Flat belief tracking is sound and complete for **every formula**
- ▶ Time and space complexity is **exponential** in # of unknown variables

Example: Non-deterministic Windows with Key (Unobs.)



- windows W_1, \dots, W_n that can be open, closed, or locked
- agent doesn't know its position, windows' status, or key position
- goal is to have **all windows locked**
- when unlocked, **windows open/close non-det.** when agent moves
- to lock window: must close and then lock it with **key**
- key must be **grabbed** to lock windows
- agent is **blind**: plan repeat n $\langle \text{Grab, Fwd} \rangle$; repeat n $\langle \text{Close, Lock, Fwd} \rangle$

Example: Non-deterministic Windows with Key (Unobs.)



Variables:

- Windows' status: $W_i \in \{open, closed, locked\}$
- Position of agent $Loc \in \{1, \dots, n\}$ and key $KLoc \in \{1, \dots, n, hand\}$

Flat belief tracking:

- single belief that initially contain $n^2 \times 3^n$ states
- each update operation (i.e. compute b_a or b_a^o) takes **exponential time**

Other Approaches for Logical POMDPs

Flat belief tracking doesn't **exploit structure** of problem

Other options for states given in terms of variables:

– as **CNF/DNF formulas**:

- ▶ **Advantage:** economic updates, succinct representation
- ▶ **Disadvantage:** intractable query answering

– as **OBDD formulas**:

- ▶ **Advantage:** tractable query answering
- ▶ **Disadvantage:** size of representation may explode quickly

– **knowledge compiled at propositional level:**

- ▶ **Advantage:** tractable in parameter called **width**
- ▶ **Disadvantage:** scope is limited to deterministic planning

Want: Factored Algorithm for Belief Tracking

Algorithm must be **general**: applicable to any instance of the model

Three key facts about **dynamic of information** in planning:

- don't need completeness for every formula. Only need to check validity of literals ' $X = x$ ' appearing in **preconditions** and **goals**
- not every variable is “correlated” to each other
- uncertainty only propagates through conditional effects of actions

Can we exploit structure and “independence” among variables?

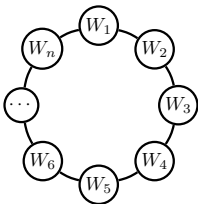
Insight!

Instead of tracking on one big problem P , track on several smaller **subproblems** P_X (simultaneously)

Hopefully, largest subproblem will be **much smaller** than P

Combined complexity: # subproblems \times complexity largest P_X

Example: Non-deterministic Windows with Key (Unobs.)



Subproblems:

- **One subproblem P_i for each window W_i**
- Subproblem P_i involves only the variables W_i , Loc and KLoc
- Flat belief tracking is done in **parallel and independently** over all subproblems

Usage:

- Queries about window W_i are answered by **inspecting belief for subproblem P_i**

Result:

- **Sound and complete** belief tracking for planning
- **Combined time/space complexity:** $O(n^3)$ for n windows

Key Idea: Decompositions

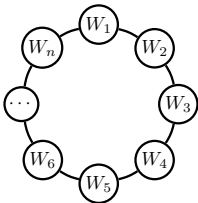
A decomposition of problem P is pair $D = \langle T, B \rangle$ where

- T is subset of **target** variables, and
- **contexts** $B(X)$ for X in T is a subset of state variables

Decomposition $D = \langle T, B \rangle$ decomposes P into subproblems:

- one subproblem P_X for each target variable X in T
- subproblem P_X involves only state variables in $B(X)$

Example: Non-deterministic Windows with Key (Unobs.)



Decomposition $D = \langle T, B \rangle$ where:

- $T = \{W_1, W_2, \dots, W_n\}$ (target variables are window's status variables)
- $B(W_i) = \{W_i, \text{Loc}, \text{KLoc}\}$ for each $i = 1, \dots, n$
- that is, total of n subproblems P_i with 3 variables each

Result:

- belief tracking over all subproblems gives **sound and complete algorithm**
- flat belief tracking on original problem has **exponential complexity** $O(n^2 3^n)$
- flat belief tracking on all subproblems has **combined complexity** $O(n^3)$

Factored Decomposition

Decomposition $F = \langle T_F, B_F \rangle$ where:

- target variables T_F are those in preconditions and goal
- contexts $B_F(X)$ given by variables Y **relevant** to X

Relevance relation captures:

- **causal relevance** induced by conditional effects and sensing formulas
- **evidential relevance** induced by observables and causal chains

Akin to relevance notions in Bayesian networks!

Factored Decomposition

Decomposition $F = \langle T_F, B_F \rangle$ where:

- target variables T_F are those in preconditions and goal
- contexts $B_F(X)$ given by variables Y **relevant** to X




Theorem

*Belief tracking over factored decomposition is **sound and complete**, and exponential in the **width** of the problem*






Width of problem:

*max number of **unknown** state variables that are all **relevant** to a given precondition or goal variable X*

Example: Wumpus and Minesweeper

Stench		Breeze	PIT
	Breeze Stench 	PIT	Breeze
Stench		Breeze	
	Breeze	PIT	Breeze

Wumpus

	2		
	3		4
1	3		2
1		2	1

Minesweeper

Factored belief tracking: exponential in **width** which is $O(n)$ for n cells

Can't do tractable tracking on these due to the large width! 😞

New Decomposition: Causal Decomposition

Decomposition $C = \langle T_C, B_C \rangle$ where:

- target variables T_F are precondition, goal **and observable variables**
- contexts $B_C(X)$ given by variables Y **causally relevant** to X




Theorem

*Belief tracking over causal decomposition is **sound**, and exponential in the **causal width** of the problem*






Causal width of problem:

*max number of **unknown** state variables that are all **causally relevant** to a given precondition, goal or observable variable X*

Example: Wumpus and Minesweeper

Stench		Breeze	PIT
	Breeze Stench 	PIT	Breeze
Stench		Breeze	
	Breeze	PIT	Breeze

Wumpus

	2		
	3		4
1	3		2
1		2	1

Minesweeper

Factored belief tracking: exponential in **width** which is $O(n)$ for n cells

Causal belief tracking: exponential in **causal width** which is

– Wumpus: **constant** 4 for any number of cells 😊

– Minesweeper: **constant** 9 for any number of cells 😊

Incompleteness too strong on these problems for solution! 😞

Complete Belief Tracking over Causal Decomposition

Tracking over causal decomposition is **incomplete** because:

- two beliefs b_X and b_Y associated with target variables X and Y may interact and are not independent

Algorithm made complete by **enforcing consistency** among local beliefs

Resulting algorithm is:

- complete for the class of **causally decomposable problems** 😊
- space exponential in **causal width** 😊
- time exponential in **width** 😞

Things are better but still not practical ...

Effective Tracking over Causal Decomposition: Beam Tracking

Replaces costly enforcement of consistency by **effective approximation**

Beam tracking is:

- time and space exponential in **causal width** 😊
- sound and powerful, but not complete
- **practical algorithm** as it is general and effective 😊

Demo

Domains:

- Minesweeper
- Wumpus: static and non-deterministic
- Battleship

On all these domains:

- crucial task is belief tracking
- action selection is **online** done w/ 1-step lookahead and simple heuristics
- **match/exceed quality** of (handcrafted) state-of-the-art controllers
- **run 2-3 orders of magnitude faster** than state of the art

Discussion

Related Work

Belief tracking “compiled” at propositional level inside planning problem:

- Det. conformant planning [Palacios & Geffner, JAIR 2009]
- Det. contingent planning [Albore et al., IJCAI 2009; B & Geffner, IJCAI 2011, Shani & Brafman, IJCAI 2011; Brafman & Shani, AAAI 2012]

Belief tracking using non-flat representations:

- logical filtering [Amir & Russell, IJCAI 2003]
- OBDDs [Cimatti et al., AIJ 2004]
- CNF [Hoffmann & Brafman, ICAPS 2005, AIJ 2006]
- DNF/CNF [To et al., IJCAI 2011]

Belief tracking on probabilistic models:

- Kalman filter (strong assumptions; fundamental in control theory)
- Hidden Markov Models (flat) / Dynamic Bayesian Networks (factored)
- particle filters (widespread use; technical and practical difficulties)

Conclusions

- **Planning is model-based approach for autonomous behaviour**
- Main challenge in planning is to achieve **generality and scalability**
- Progress continuously assessed in **benchmarks and competitions**
- Planning with sensing is **belief tracking** plus **action selection**
- Three factored algorithms for belief tracking:
 - ▶ Factored BT: sound and complete; exponential in **width**
 - ▶ Causal BT: sound but weak; exponential in **causal width**
 - ▶ Beam tracking: sound and effective; exponential in **causal width**

Challenges

- Effective action selection for planning with sensing isn't clear yet
 - ▶ algorithms + heuristics (or base policies)
- Deployment of these methods for other AI models
- Probabilistic belief tracking; applications like robotics; SLAM; ...

More Information

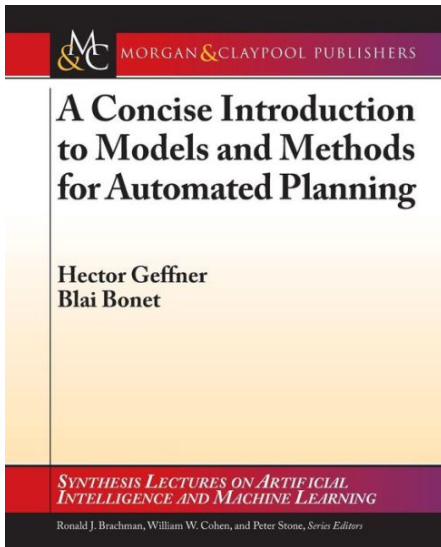
Papers, slides, other groups, etc.:

- My page: <http://www ldc.usb.ve/~bonet>
- Hector Geffner's page: <http://www.tecn.upf.es/~hgeffner>
- Ronen Brafman's page: <http://www.cs.bgu.ac.il/~brafman>

Software:

- Belief tracking: <http://code.google.com/p/belief-tracking>
- K-replanner: <http://code.google.com/p/cp2fsc-and-replanner>
- Software for solving MDPs: <http://code.google.com/p/mdp-engine>
- Other: see my page

New Book on AI Planning



Thanks. Questions?