Automated Action Planning

Classical Planning for Non-Classical Planning Formalisms

Carmel Domshlak

Automated Action Planning

— Classical Planning for Non-Classical Planning Formalisms

Beyond Classical Planning

Richer models people are working on

1. Temporal Planning (action have duration)
2. Metric Planning (continuous variables)
3. Planning with Preferences
4. Planning with Resource Constraints
5. Net-benefit Planning (maximize net value of goals achieved)
6. Generalized Planning (complex control structures, such as loops)
7. Multi-agent Planning
8. Planning Under Uncertainty:
    - 8.1 Conformant Planning
    - 8.2 Contingent Planning
    - 8.3 Markov Decision Processes (MDPs)
    - 8.4 Partially Observable MDPs
    - 8.5 Conformant Probabilistic Planning (Fully Unobservable POMDPs)

How many courses on planning do we need?

Key Insights:

- ☺ Classical planning offers a wealth of ideas for generating good solutions, fast.
- ☹ Importing these ideas to each of the above non-classical formalisms is difficult, and often simply does not work.

Yet:

- ☺ Goal oriented sequencing of actions is a fundamental computational problem at the heart of all planning problems.
- ☺ Classical planners have reached a certain performance level that makes them attractive for addressing this problem.

So...

Two Strategies

1. Top-down:
   Develop native solvers for more general class of models
   - $+$: generality
   - $-$: complexity
2. Bottom-up: Extend the scope of 'classical' solvers
   - $+$: efficiency
   - $-$: generality

We now explore the second approach

Using Classical Planners within Non-Classical Planners

Two Key Techniques:

1. Replanning: the classical problem is an optimistic view of the original problem
2. Compilation: the classical problem is equivalent to the original problem
   (possibly under certain reasonable conditions)

Replanning

An online method for solving planning problems with some uncertainty

1. Make some assumptions $\rightarrow$ get a simpler model
2. Solve simpler model
3. Execute until your observation contradict your assumptions
4. Repeat (Replan)

An established technique:

- Underlies many closed loop controllers
- Used in motion planning under uncertainty

Motivation: Why Analyzing the Expressive Power?

- Expressive power is the motivation for designing new planning languages

⇝ Often there is the question: *Syntactic sugar* or *essential feature*?

- *Compiling away* or change planning algorithm?
- If a feature can be compiled away, then it is apparently only *syntactic sugar*.
- However, a compilation can lead to much larger planning domain descriptions or to much longer plans.

⇝ This means the planning algorithm will probably choke, i.e., it cannot be considered as a compilation

Example: DNF Preconditions

- Assume we have **DNF preconditions** in STRIPS operators
- This can be **compiled away** as follows
- Split each operator with a DNF precondition $c_1 \vee \ldots \vee c_n$ into n operators with the same effects and c_i as preconditions

$\leadsto$ If there exists a plan for the original planning task there is one for the new planning task and *vice versa*

$\longrightarrow$ The planning task has almost the same size

$\longrightarrow$ The shortest plans have the same size

Example: Conditional effects

- Can we compile away **conditional effects** to STRIPS?
- Example operator: $\langle a, b \rhd d \wedge \neg c \rhd e \rangle$
- Can be translated into four operators:
  $\langle a \wedge b \wedge c, d \rangle, \langle a \wedge b \wedge \neg c, d \wedge e \rangle, \ldots$
- Plan existence and plan size are identical
- Exponential blowup of domain description!

$\longrightarrow$ Can this be avoided?

FDR Planning with Soft Goals

- Planning with soft goals aimed at plans π that maximize utility

$$u(\pi) = \sum_{p \in app_{\pi}(I)} u(p) \quad - \quad \sum_{a \in \pi} cost(a)$$

- Best plans achieve best tradeoff between action costs and rewards
  $\rightsquigarrow$ Note: "do nothing" is always a valid plan.
  $\rightarrow$ Suggests conceptual difference?
- Model used in recent planning competitions; net-benefit track 2008 IPC
- Yet soft goals do not add expressive power; they can be compiled away

FDR Planning with Soft Goals

- For each soft goal p, create new hard goal p' initially false, and two new actions:
  - $collect(p)$ with precondition p, effect p' and cost 0, and
  - $forgo(p)$ with an empty precondition, effect p' and cost $u(p)$
- Plans π maximize $u(\pi)$ iff minimize $cost(\pi) = \sum_{a \in \pi} cost(a)$ in resulting problem
- Any helpful in practice?
- Compilation yields better results that native soft goal planners in 2008 IPC [KG07]

	IPC-2008 Net-Benefit Track			Compiled Problems			
Domain	Gamer	HSP^*_{P}	Mips-XXL	Gamer	HSP^*_{F}	HSP^*_{0}	Mips-XXL
crewplanning(30)	4	16	8	-	8	21	8
elevators (30)	11	5	4	18	8	8	3
openstacks (30)	7	5	2	6	4	6	1
pegsol (30)	24	0	23	22	26	14	22
transport (30)	12	12	9	-	15	15	9
woodworking (30)	13	11	9	-	23	22	7
total	71	49	55		84	86	50

Temporal Planning –
Compilation to Classical Planning

Antonín Komenda

slides based on Crikey 3 slides

February 27, 2017

Planning with Time

- classical planning has instantaneous actions (no explicit duration)
  - **preconditions**→action→**effects**
- temporal planning has actions with durations (from PDDL2.1)
  - **start conditions**→action start→start effects
  - *duration* of the action, over all *condition* (invariant)
  - end preconditions→action end→**end effects**

Durative Actions in PDDL

```
(:durative-action LOAD-TRUCK
  :parameters (?obj - obj ?truck - truck ?loc - location)
  :duration (= ?duration 2)
  :condition (and
    (over all (at ?truck ?loc))
    (at start (at ?obj ?loc)) )
  :effect (and
    (at start (not (at ?obj ?loc)))
    (at end (in ?obj ?truck))
  )
)
```

Action Compilation

- compilation of the durative actions to STRIPS
- solve the STRIPS problem
- reconstruct the temporal plan?

Action Compression

- firstly used in TGP planner
- the TGP compilation removes the distinction of start and end parts of durative actions
  - preconditions = start condition $\wedge$ end condition $\wedge$ over all condition
  - effects = start effects $\wedge$ end effects
- is this enough? is temporal planning syntactic sugar?

Required Concurrency

- No.
- TGP compilation is *unsound* and *incomplete*

The Match Problem

- Consider:
  - An engineer must mend a fuse in a dark cellar
  - To do this he will require light, which can be provided by a match
  - He can perform two actions: light a match and mend a fuse
- durative actions: LIGHT_MATCH 8s, MEND_FUSE 5s
- LIGHT_MATCH needs an unused match u at the beginning and lights l the match over all
- LIGHT_MATCH uses the match $\neg u$ and lights the match l at the beginning
- LIGHT_MATCH blows out the light $\neg l$ and not use it u at the end
- MEND_FUSE needs light l over all duration (and free hands f)
- MEND_FUSE mends the fuse m at the end (hands are not free $\neg f$ at the beginning, but are free at the end f)

LPGP Compilation (on Example)

- LIGHT_MATCH_START pre:u, eff:$\neg u, l, z$
  (new atom z – action started)
- LIGHT_MATCH_INV pre:z, l, eff: i
  (new atom i – inv. checked)
- LIGHT_MATCH_END pre: z, i, eff: $\neg l, \neg z, \neg i$
- Problems:
  - we need to ensure that all actions that are started have ended
  - invariants can be violated

CRIKEY!

- Solution: Crikey! planner
- using Simple Temporal Networks (STN)
- condition whether all actions ended in goal
- keeps scheduling constraints
  - at each state builds STN
  - uses Floyd-Warshall to check negative temporal cycles, if such exist the STN is inconsistent
  - prune stated with inconsistent STNs