Heuristics for Planning with Penalties and Rewards Using Compiled Knowledge

Blai Bonet
Universidad Simón Bolívar
Caracas, Venezuela

Héctor Geffner
ICREA & Univ. Pompeu Fabra
Barcelona, Spain

Motivation

- Planning is a form of **general problem solving**

$$\textit{Problem} \Longrightarrow \textit{Language} \Longrightarrow \boxed{\textit{Planner}} \Longrightarrow \textit{Solution}$$

- **Idea:** problems **described** at high-level and **solved** automatically
- **Goal:** facilitate modeling with small penalty in performance

Planning and General Problem Solving: How general?

For which **class of problems** planner should work?

- **Classical planning** focuses on problems that map into **state models**
  - a state space S
  - an initial state $s_0 \in S$
  - goal states $S_G \subseteq S$
  - actions $A(s)$ applicable in each s
  - a successor state function $f(a, s)$, $a \in A(s)$
  - action costs $c(a, s) = 1$
- The **solution** of this model is an applicable action sequence that maps s_0 into a goal state
- A solution is **optimal** if it minimizes the sum of action costs

Variety of Models in Planning

- Other forms of planning work over different models; e.g. **conformant planning** works over models given by
  - a state space S
  - an initial **set of states** $S_0 \in S$
  - goal states $S_G \subseteq S$
  - actions $A(s)$ applicable in each s
  - a **set** of possible succesor states $F(a, s)$, $a \in A(s)$
  - action costs $c(a, s) = 1$
- If model extended with **sensors**, we get model for **contingent planning**,
- If uncertainty quantified with probabilities, we get **MDPs** and **POMDPs**

A more precise definition of Planning

- **Planning** is about development of **solvers** for certain classes of **models**

- The **models** expressed in compact form over **planning languages**

- For example, in **Strips**, a 'classical planning problem' expressed as tuple $\langle F, O, I, G \rangle$ where

  – F stands for set of all **fluents** or **atoms** (boolean vars)
  – O stands for set of all **actions**
  – $I \subseteq F$ stands for **initial situation**
  – $G \subseteq F$ stands for **goal situation**

  and each action a represented by

  -- **Add** list $Add(a) \subseteq F$
  -- **Delete** list $Del(a) \subseteq F$
  -- **Precondition** list $Pre(a) \subseteq F$

From Language to Model: Semantics of Strips

Strips problem $P = \langle F, O, I, G \rangle$ represents **state model** $\mathcal{S}(P)$

- the states $s \in S$ are **collections of atoms**
- the initial state s_0 is I
- the goal states $s \in S_G$ are such that $G \subseteq s$
- the actions in s are the $a \in O$ s.t. $Pre(a) \subseteq s$
- the state that results from doing a in s is $s' = s - Del(a) + Add(a)$
- **action costs** $c(a, s)$ are all 1

The (optimal) **solution** of planning problem P is the (optimal) solution of State Model $\mathcal{S}(P)$

Progress in Classical Planning

- **large problems solved fast**
- **empirical methodology**
  - standard PDDL language (richer than Strips)
  - planners and benchmarks available
  - focus on performance, planning competitions, . . .
- **novel ideas and formulations**
  - e.g., extraction and use of **heuristics** $h(s)$ for guiding **search**

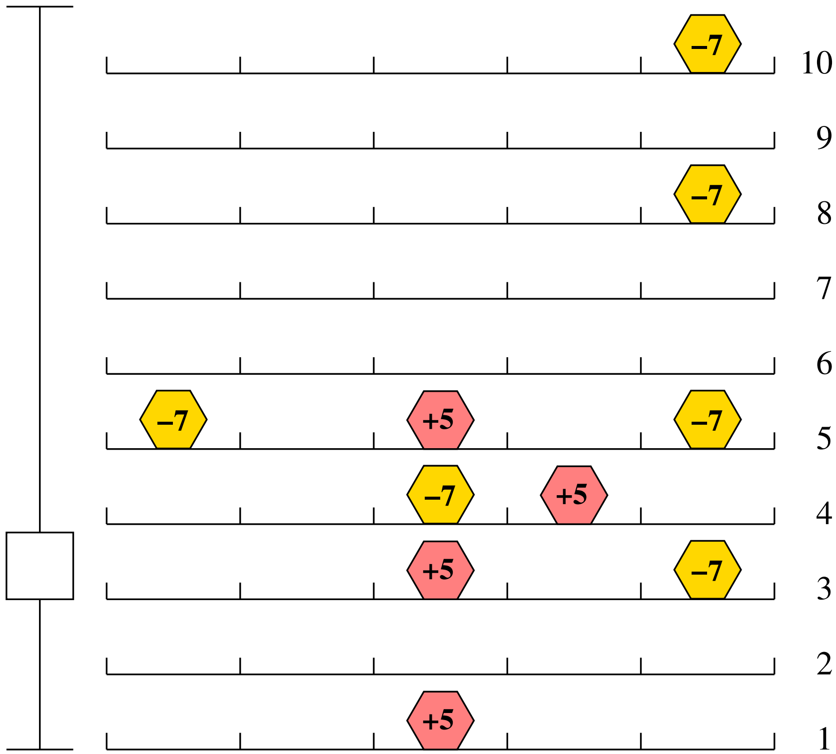
Our goal in this work

Extend classical planning methods to richer cost model

$$c(a,s) = \begin{cases} c & \text{uniform costs} \\ c(a) & \text{action-dependent costs} \\ c(s) & \text{state-dependent costs} \end{cases}$$

- We want to be able to plan with **state-dependent costs** which may be **positive (penalties)** or **negative (rewards)**
- For this, we will a define a **richer cost model** and **heuristic** h_c^+

Planning with Penalties and Rewards: Example



- Elevator Problem with 10 floors, 5 positions, 1 elevator.
- **No hard goals: penalties** and **rewards** associated with positions
- **Question:** How to model and solve these problems effectively?

Some Results using New Heuristic h_c^+

Elevator instance n-m-k has n floors, m positions, and k elevators

Problem	Length	Optimal Cost	Solved with h_c^+		Solved blind	
			Time	Nodes	Time	Nodes
4-4-2	12	-9	0.35	1,382	4.19	29,247
6-6-2	23	-14	21.44	24,386	2,965.90	6,229,815
6-6-3	23	-14	133.48	76,128	---	---
10-5-1	11	-3	0.39	238	161.85	445,956
10-5-2	32	-5	330.72	189,131	---	---

- What is the **cost model**, and how **heuristic** h_c^+ defined, computed, and used in the search?
- Heuristic h_c^+ not only must estimate cost to goal, but **must select the goals too!**

Where is the KR?

- We show how to construct a **circuit** whose input is a state s, and whose output, computed in **linear time**, is $h_c^+(s)$ for **any** s

- For this, the **heuristic** $h_c^+(s)$ is formulated as **rank** $r(T(P) \wedge I(s))$ of propositional theory $T(P) \wedge I(s)$ obtained from problem P and state s

$$r(T) = \min_{M \models T} r(M) \quad \text{and} \quad r(M) = \sum_{L \in M} r(L)$$

- Rank $r(T(P) \wedge I(s))$ **intractable** in general but computable in **linear time** if $T(P)$ **compiled** into d-DNNF (Darwiche)

- The **circuit** is the compiled $T(P)$ formula; the compilation may take exponential time and space, but not necessarily so (like OBDDs)

Our plan for (the rest!) of the talk

1. Define the **cost model** for planning problems P
2. Define the **heuristic** h_c^+ as relaxation of model
3. Define **encoding** $T(P)$ such that $h_c^+(s) = r(T(P) \wedge I(s))$
   - $T(P)$ *defined in terms of 'strong completion' of a Logic Program* P'*; so* $h_c^+(s)$ *can also be thought as* **rank of best answer set** *of* P'
4. Define heuristic search **algorithm** (Dijkstra, A*, IDA*, etc don't work with **negative** heuristics and costs!)
5. Present experimental results

Planning Model

- A **problem** P expressed in planning language extended with **positive action cost** $c(a)$ and **positive** or **negative fluent costs** $c(p)$

- **Cost of a plan** π for P given by cost of the actions in π and the atoms $F(\pi)$ made true by π (at **any** time)

$$c(\pi) \ \stackrel{\text{def}}{=} \ \sum_{a \in \pi} c(a) \ + \ \sum_{p \in F(\pi)} c(p)$$

- **Cost of problem** P is

$$c^*(P) = \min_{\pi} c(\pi)$$

Heuristic h_c^+

- **Heuristic** $h_c^+(P)$ defined in terms of the **delete-relaxation** P^+:

$$h_c^+(P) \stackrel{\text{def}}{=} c^*(P^+)$$

- Heuristic h_c^+ is **informative** and **admissible** (under certain conditions)

- For the **classical cost function** $c(action) = 1$ and $c(fluents) = 0$, h_c^+ is the well known delete-relaxation heuristic, **approximated by tractable heuristics** in planners such as HSP and FF

Modeling

The model is simple but flexible, and can represent . . .

- **Terminal Costs:** an atom p can be rewarded or penalized if true at the end of the plan, by means of new atom p' initialized to false, and conditional effect $p \rightarrow p'$ for action End.
- **Goals:** not strictly required; can be modeled as a sufficiently high terminal reward
- **Conditional Preferences:** in terms of conditional effects
- **Rewards on Conjunctions:** in terms of atoms and actions . . .

Not so simple to represent **repeated costs or rewards**, **penalties on sets of atoms** (would need ramifications), **partial preferences**, . . .

Heuristics, Ranks and d-DNNF Compilation

Claim: If $h_c^+(s) = r(T(P) \wedge I(s))$ where $I(s)$ is a set of literals and

$$r(T) = \min_{M \models T} r(M) \quad \text{and} \quad r(M) = \sum_{L \in M} r(L)$$

then $h_c^+(s)$ computable in linear linear **for any** s if $T(P)$ in d-DNNF.

This follows from two results by **Darwiche and Marquis** about d-DNNF:

1. **Ranks:** If T in d-DNNF then $r(T)$ computable in **linear time**
2. **Conjoining:** If T in d-DNNF and I is a set of literals, then $T \wedge I$ can be brought into d-DNNF in **linear-time** too

Stratified Encodings

Plans for a Strips problem $P = \langle F, I, O, G \rangle$ with horizon n can be obtained from **models** of propositional theory $T_n(P)$ (Kautz and Selman):

1. **Actions:** For $i = 0, 1, \ldots, n-1$ and all a
   $a_i \supset p_i$ for $p \in Pre(a)$
   $C_i \wedge a_i \supset L_{i+1}$ for each effect $a : C \rightarrow L$

2. **Frame:** For $i = 0, \ldots, n-1$ and all p
   $p_i \wedge (\bigwedge_{a:C \rightarrow \neg p} (\neg a_i \vee \neg C_i)) \quad \supset p_{i+1}$
   $\neg p_i \wedge (\bigwedge_{a:C \rightarrow p} (\neg a_i \vee \neg C_i)) \quad \supset \neg p_{i+1}$

3. **Seriality**, **Init**, **Goals**, . . .

Heuristic h_c^+ could be defined from suitable rank of theory $T_n(P^+)$, where P is the delete-relaxation, yet . . .

- how to define the **horizon** n and deal with large n?
- how to define **ranking** so that $h_c^+(s) = r(T_n(P^+) \wedge I(s))$?

Logic Program Encodings: Implicit Stratification

- **Normal Actions:** For each **positive** (conditional) effect $a : C \rightarrow p$ of action a with precond $Pre(a)$ add

$$p \leftarrow C, Pre(a), a$$

- **Set Actions:** Add $set(p)$ actions, which are **true** in $I(s)$ iff $p \in s$:

$$p \leftarrow set(p)$$

Let $T(P)$ be resulting **logic program**, and $\mathrm{wffc}(T(P)$ be the formula that picks up the models of $T(P)$ where fluents are **well-supported**:

Theorem: For any s, $h_c^+(s) = r(\mathrm{wffc}(T(P)) \wedge I(s))$, where r is the **literal** ranking function s.t $r(l) = c(l)$ for positive literals l and $r(l) = 0$ otherwise.

Main Theorem

If the theory $\mathrm{wffc}(T(P))$ is compiled into d-DNNF, then the value $h_c^+(s)$ can be computed for **any state** s **and cost function** c **in linear time.**

Some remarks:

- The compilation of $\mathrm{wffc}(T(P))$ may take exponential time and space, although this is not necessarily so (like OBDDs)
- The search for plans requires computing $h_c^+(s)$ at **many states**; effort of compilation amortized throughout these intractable calls
- Similar ideas can be used for deriving the heuristic values $h_c^+(g)$ for **any subgoal** g for guiding a **regression** search.
- Last, admissible approximations of h_c^+ can be obtained by 'relaxing' the problem (e.g., removing non-rewarding atoms) . . .

From Heuristic to Search

- A* does not work due to negative edge costs and heuristics
- However, since heuristic is **monotone**, only need to change **termination condition**
- In addition, due to semantics of model, nodes in search graph must **keep track of penalties and rewards collected**

Empirical Results: Compilation Serialized Logistics

Problem	backward theory Time	backward theory Nodes	forward theory Time	forward theory Nodes
4-0	0.34	1,163	1.66	64,623
...	....	.....	....	......
6-2	0.21	1,163	1.64	63,507
6-3	0.32	1,163	1.65	64,951
7-0	1.26	3,833	145.83	3,272,308
7-1	1.38	3,837	142.82	3,211,456
8-0	1.30	3,833	263.20	3,268,023
8-1	1.37	3,837	263.19	3,270,570
9-0	1.98	3,854	147.82	3,199,190
9-1	1.27	3,833	138.81	3,130,689
10-0	6.86	13,153	---	---
10-1	6.87	13,090	---	---

- These are first 18 logistic problems from 2nd IPC (serialized)
- d-DNNF compiler due to Darwiche (c2d) and Completion $\mathrm{wffc}(T)$ obtained following Lin and Zhao, IJCAI-03.

Runtime Serialized Logistics

Problem	$c^*(P)$	h^2 backward			h_c^+ with mutex backward		
		$h^2(P)$	Time	Nodes	$h_c^+(P)$	Time	Nodes
4-0	20	12	0.23	4,295	19	0.02	40
...	..	..	....	.....	..	....	...
6-2	25	10	25.49	301,054	23	0.89	517
6-3	24	12	7.87	99,827	21	0.84	727
7-0	36	12	---	---	33	97.41	4,973
7-1	44	12	---	---	39	4,157.70	175,886
8-0	31	12	---	---	29	11.64	591
8-1	44	12	---	---	41	283.32	12,913
9-0	36	12	---	---	33	65.81	3,083
9-1	30	12	---	---	29	1.54	81
10-0	?	12	---	---	41	---	---
10-1	42	12	---	---	39	5,699.2	20,220

- Heuristic h^2 planner corresponds basically to 'serial' Graphplan
- Heuristic h_c^+ with mutex, adds $h_c^+(g) = \infty$ when g **mutex**

Elevator

Elevator instance n-m-k has n floors, m positions, and k elevators

Problem	Length	Optimal Cost	Solved with h_c^+ Time	Nodes	Solved blind Time	Nodes
4-4-2	12	-9	0.35	1,382	4.19	29,247
6-6-2	23	-14	21.44	24,386	2,965.90	6,229,815
6-6-3	23	-14	133.48	76,128	---	---
10-5-1	11	-3	0.39	238	161.85	445,956
10-5-2	32	-5	330.72	189,131	---	---

- Theory $\mathrm{wffc}(T(P))$ does not actually compile for **Elevator**
- Heuristic above is **admissible approximation** that results from 'relaxing' atom $inside(e)$ from P

Blocks

- Block World instances do not compile as well as Logistics
- We could only compile first 8 instances from the 2nd IPC
- These are very small instances having at most 6 blocks, where h_c^+ does not pay off (for classical planning)

Wrap up

In this work we have combined ideas from a number of areas, such as **search**, **planning**, **knowledge compilation**, and **answer set programming** to define and compute an **heuristic for optimal planning with penalities and rewards**

- Some theories compile well, others do not; in certain cases, admissible and informed **approximations** obtained from 'relaxing' certain atoms
- Correspondence between **heuristic** and **rank** of **preferred models** or **answer sets** suggests possible use of **Weighted SAT** or **ASP solvers**
- Compilation-based approach, however, yields *circuit* or *evaluation network* that maps **situations** into **appraisals** in linear-time; a role similar to **emotions** . . .