Heuristic Search Planning: Overview and Prospects

Hector Geffner
Universidad Simón Bolívar
Caracas, Venezuela
9/2000

Motivation

- Development of **General Problem Solvers** has been a main goal in Artificial Intelligence

Problem $\longrightarrow$ *Language* $\longrightarrow$ $\boxed{\begin{array}{c}\textit{Representation}\\ +\\ \textit{Algorithms}\end{array}}$ $\longrightarrow$ *Solution*

- Ideally one would describe problem at high-level, and computer would take care of the rest
- Example problems

8-puzzle	*rubik*	*mastermind*	*12 coins*
diagnosis	*scheduling*	*tsp*	*robot navig*
sorting	*minesweeper*	*crypto-arith*	*vehicle routing*
...	...		

Framework

how to make sense of this variety of problems?

Three distinguished components:

- **Representation languages** for describing problems conveniently
- **Mathematical models** for making sense of classes of problems
- **Algorithms** for solving these models

Mathematical Models: State Models

State models are very common in AI and determined by

- finite and discrete state space S
- an initial state $s_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each state $s \in S$
- a transition function $f(a, s)$ for $s \in S$ and $a \in A(s)$
- action costs $c(a, s) > 0$

– A *solution* for this model is a sequence of applicable actions that maps the initial state s_0 into a goal state $s \in S_G$

– Problems like TSP, Sliding puzzles, and Rubik's cube can be naturally modeled and solved as **state models**

– Problems like Mastermind, Robot Navigation, Scheduling, and others do not

Some Mathematical Models in Planning

- State Models (SMs)
  - *simplest type of models*
- Non Deterministic and Probabilistic State Models
  - *more complex* **dynamics**
- Partially Observable Non-Det and Prob State Models
  - *more complex sensing (***feedback***)*
- Scheduling/Temporal Models
  - **time**, *action durations, concurrency, resources*

Focus in this talk: Classical Planning

- **Classical planning** deals with representation and solution of problems that can be formulated as **State Models**
- **Main ingredients** are
  - **languages** for representing State Models compactly (e.g., Strips, ADL, ...)
  - **algorithms** for solving State Models that exploit the compact representations

The Strips Language (Fikes and Nilsson 71)

- **Strips** is one of the oldest, simplest, and most used planning languages, even if **not too flexible**
- A **problem** in Strips is a tuple $\langle A, O, I, G \rangle$ where
  - A stands for set of all **atoms** (boolean vars)
  - O stands for set of all **operators** (ground actions)
  - $I \subseteq A$ stands for **initial situation**
  - $G \subseteq A$ stands for **goal situation**
- Operators $o \in O$ **represented** by three lists
  - the **Add** list $Add(o) \subseteq A$
  - the **Delete** list $Del(o) \subseteq A$
  - the **Precondition** list $Pre(o) \subseteq A$
- Intuitively, $Pre(o)$ encodes the atoms that must be true for o to be applicable, $Add(o)$ encodes the atoms that o makes true, and $Del(o)$ encodes the atoms that o makes false

Semantics of Strips: From Strips to State Models

Strips problem $P = \langle A, O, I, G \rangle$ determines **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 $op \in O$ s.t. $Prec(op) \subseteq s$
- the state that results from doing op in s is $s' = s - Del(op) + Add(op)$
- action costs $c(op, s)$ are all 1

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

Algorithms for Solving State Models

Planning problems P could be solved by standard **search algorithms** over state space $\mathcal{S}(P)$, yet this approach not pursued successfully until recently ...

- **Blind search/Brute force algorithms**
  - Goal plays **passive** role in the search

    e.g., Depth First Search (DFS), Breadth-first search (BrFS), Uniform Cost (Dijkstra), Iterative Deepening (ID)
- **Informed/Heuristic Search Algorithms**
  - Search uses a function $h(s)$ that estimates 'distance' (cost) from state s to S_G to guide search

    e.g., A, IDA*, Hill Climbing, Best First Search (BFS), Branch & Bound*

Approaches to Domain-Independent Planning

- **Strips algorithm** (70's): Total ordering planning backward from Goal; work always on **top** subgoal in stack, delay rest
- **Partial Order (POCL) Planning** (80's): work on **any** subgoal, resolve threats; UCPOP 1992
- **Graphplan** (1995 – ...): build graph containing all possible **parallel** plans up to certain length; then extract plan by searching the graph backward from Goal
- **SatPlan** (1996 – ...): map planning problem given horizon into SAT problem; use state-of-the-art SAT solver
- **Heuristic Search Planning** (1997 – ...): search state space $\mathcal{S}(P)$ with heuristic function h extracted from problem P
- **Model Checking Planning** (1998 – ...): search state space $\mathcal{S}(P)$ with 'symbolic' Breadth First Search where sets of states represented by formulas implemented by BDDs

Key ingredients in Heuristic Search Planning

- **State Space**
  - progression space (forward planning)
  - regression space (backward planning)
- **Search Algorithms**
  - Hill-Climbing, A*, Weighted A*, IDA*, ...
- **Heuristics**
  - extracted automatically from representation; it's the most **critical** component in heuristic search planning and what distinguishes it from specialized problem solving.

Some Heuristic Search Planners

1. UNPOP (McDermott)
2. HSP (Bonet & Geffner)
3. GRT (Refanidis & Vlahavas)
4. FF (Hoffman)
5. MIPS (Edelkamp ; also uses other techniques)
6. STAN (Fox & Lang ; also uses other techniques)

Planners 2–6 entered AIPS-2000 Planning Contest where they were the top 5 performers (among approx. 13 planners)

Top performing planner was FF (Fast Forward)

Heuristic Functions

- Derived as **optimal** cost function of **relaxed problem**

  e.g., Manhattan distance heuristic is optimal function of puzzle in which tiles are allowed to move into **any** neighboring position

- In planning, simple relaxation can be obtained by **ignoring Delete Lists**

- Resulting heuristic is informative and admissible ... but is still **intractable**

Additive Heuristic Function

- Heuristic used in UNPOP and HSP makes further assumption that **subgoals are independent**
- As a result, the cost of **sets** of atoms (subgoals) is given by **sum** of the costs of each of atom in the set
- For all **atoms** p:

$$g(p;s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } p \in s \\ \min_{a \in O(p)} [1 + g(Prec(a);s)] & \text{otherwise} \end{cases}$$

- For **sets** of atoms C:

$$g(C;s) \stackrel{\text{def}}{=} \sum_{r \in C} g(r;s)$$

- Resulting heuristic h_{add}

$$h_{add}(s) \stackrel{\text{def}}{=} g(Goals;s)$$

is **informative** but is **not admissible** . . .

Admissible Max heuristic h_{max}

Assumes that cost of **sets** of atoms (subgoals) is given by **cost of most costly atom in set**

- For all **atoms** p:

$$g(p;s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } p \in s \\ \min_{a \in O(p)}[1 + g(Prec(a);s)] & \text{otherwise} \end{cases}$$

- For **sets** of atoms C:

$$g(C;s) \stackrel{\text{def}}{=} \max_{r \in C} g(r;s)$$

- Resulting heuristic

$$h_{max}(s) \stackrel{\text{def}}{=} g(Goals;s)$$

is **admissible** but **not very informative** ...

Heuristic underlying Graphplan

Graphplan can be understood as an Heuristic Search Planner

- **State space:** *regression space*
- **Search Algorithm:** *version of IDA**
- **Heuristic Function** $h_G(s)$*:*

  given by index of first layer in plan graph that contains the atoms in s without a mutex

- Graphplan heuristic h_G closely related to h_{max}

  $$h_{max}(s) \;\approx\; h_G(s) - \text{mutexes}$$

  i.e., h_G becomes h_{max} when mutex info ignored

- h_G **admissible** for both sequential and parallel planning, and **dominates** h_{max} (more informative)

New family of heuristics h^m

- New class of **admissible** and **polynomial** heuristics h^m, $m = 1, 2, \ldots$ for sequential and parallel planning that generalize 'max' and Graphplan heuristics (Haslum & Geffner, AIPS'00)
  - *For $m = 1$, sequential $h^m = h_{max}$*
  - *For $m = 2$, parallel $h^m = h_G$*
- **Idea:** Heuristic $h^m(C)$ approximates cost of **set** of atoms C by cost of most costly subset $D \subseteq C$, $|D| = m$
- For higher m, h^m more **accurate** but more **expensive**

Formulation of heuristics h^m

- **Optimal** h^* that estimates cost of achieving **set** of atoms C from s_0 is

$$h^*(C) = \begin{cases} 0 & \text{if } C \subseteq s_0 \\ \min_{\langle B,a\rangle \in R(C)}[cost(a) + h^*(B)] & \text{otherwise} \end{cases}$$

  where $\langle B, a\rangle \in R(C)$ if C **regressed** through a yields B

- **Heuristic** h^m that provides **admissible** estimate defined by **relaxation:**

$$h^m(C) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } C \subseteq s_0 \\ \min_{\langle B,a\rangle \in R(C)}[cost(a) + h^*(B)] & \text{if } |C| \leq m \\ \max_{D \subset C, |D|=m} \; h^m(D) & \text{otherwise} \end{cases}$$

- **Relaxation** approximates cost of large sets C, $|C| > m$, by cost of most costly subset $D \subset C$, $|D| \leq m$

- **Heuristics** h^m admissible and polynomial, and can be obtained from these equations by shortest-path algorithms

Special case 1: Max-Atom Heuristic h^1

For $m = 1$, the general equation defining h^m

$$h^m(C) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } C \subseteq s_0 \\ \min_{\langle B, a\rangle \in R(C)} [cost(a) + h^*(B)] & \text{if } |C| \leq m \\ \max_{D \subset C, |D| = m} \; h^m(D) & \text{otherwise} \end{cases}$$

becomes

$$h^m(C) = \begin{cases} 0 & \text{if } C \subseteq s_0 \\ \min_{a \in O(p)} [cost(a) + h^m(Prec(a))] & \text{if } C = \{p\} \\ \max_{p \in C} \; h^m(\{p\}) & \text{otherwise} \end{cases}$$

This is exactly the equation for h_{max}
($O(p)$ stands for the set of ops that add p)

Special case 2: Max-Pair Heuristic h^2

For $m = 2$, the equation defining $h^m(C)$ becomes

$$
\begin{array}{ll}
0 & \text{if } C \subseteq s_0 \\
\min_{a \in O(p)} [c(a) + h^m(Prec(a))] & \text{if } C = \{p\} \\
\min \big\{ \min_{a \in O(p\&q)} [c(a) + h^m(Prec(a))] \,, & \\
\quad \min_{a \in O(p|q)} [c(a) + h^m(Prec(a) \cup \{q\})], & \text{if } C = \{p, q\}; \\
\quad \min_{a \in O(q|p)} \ [c(a) + h^m(Prec(a) \cup \{p\})] \big\} & \\
\max_{\{p,q\} \in C} h^m(\{p, q\}) & \text{if } |C| > 2
\end{array}
$$

- Max-Pair Heuristic h^2 is **equivalent** to Graphplan's h_G provided action costs are equal and all actions are mutex (no parallelism)
- Formulation makes clear distinction between **definition** of the heuristic and **representations** and **procedures** used for computing it

Some results over sequential domains

HSPr* : **optimal** sequential planner = IDA* search + h^2

	STAN	BBOX	HSPr*	
Instance	Time	Time	Time	#A
blocks-9	0.4	1.0	0.45	6
blocks-11	2.0	6.9	1.29	9
blocks-15	102.7	—	136.46	14
eight-1	89.9	—	63.53	31
eight-2	62.0	—	67.23	31
eight-3	0.5	89.0	0.51	20
grid-1	1.5	14.2	8.44	14
grid-2	—	—	7:55h	26
gripper-1	0.0	0.0	0.07	3
gripper-2	0.0	0.3	0.11	9
gripper-4	2.1	*	*	*

- '—' memory exhausted
- '*' time exhausted ($>$ 12 hours)

What about parallel domains?

- HSPr* does not solve hard instances of domains like `rockets` and `logistics` that are solved by GRAPHPLAN or SAT planners
- Reason is that h^2 is **poor estimator of sequential length** in those domains
- Still, heuristics h^m can be reformulated to estimate **parallel** rather than **serial** cost, and used to guide search for **parallel optimal** plans
- The equation for **parallel** h^m for $m = 2$ just changes **min** expression in $h^m(C)$ for $C = \{p, q\}$ to

$$h_p^m(C) = \min \left\{ \ldots, \min_{\langle a, a' \rangle \in O(p,q)} [1 + h_p^2(Prec(a) \cup Prec(a'))] \right\}$$

- This additional clause allows p and q to be established by two actions in **parallel** at the cost of single action.
- Resulting heuristic is **equivalent** to Graphplan's

Optimal parallel planner HSP**r***

Problem	HSPr^*	GRAPHPLN	STAN	BBOX
rocket.a	90.5	100.0	—	1.8
rocket.b	68.6	310.0	—	2.3
log.a	11540	1235	0.4	2.1
log.b	*	579	2.0	10.4
log.c	*	—	1261	47.0

+ Problems that could not be solved **sequentially** are solved in **parallel**
- Parallel HSPr* comparable to Graphplan over rockets, but over order of magnitude slower in logistics
- Neither one competitive in these domains: problem is **branching scheme;** branching factor in **parallel/temporal** planning is too high

Branching in Search and Problem Solving

- **15-puzzle,Rubik,. . . :**
  - **Branching:** Forward/Backward ok
- **TSP (Travelling Salesman Problem)**
  - **Branching:** include/exclude edge i–j in tour
  - **Termination;** when relaxed 'assignment problem' yields single tour
- **JSP (Job Shop Scheduling)**
  - **Branching:** $i \prec j$, $j \prec i$ for pairs of tasks
  - **Termination:** when no overlap among actions requiring same resource

Branching in Planning

- **Forward:** State-space; extend plan head; totally ordered
- **Backward:** Regression-space ; extend plan tail; totally ordered
- **Temporal:** for action a and time i, create splits $a[i] =$ **true** / $a[i] =$ **false**
- **POCL:** Partial Order Causal Link Planning
  - precedence constrains $a_1 \prec a_2$ (promotion/demotion) to elim 'threats'
  - causal links $a_1 \rightarrow_P a_2$ to 'support' preconds
  - **termination:** when no threats & all preconds supported
  - .....

Future: Alternative Branching in Heuristic Search Planning

- **Idea:** Construct plans neither from **head** or **tail**, but by making partial **commitments**
- A 'state' (**partial plan**) σ is a **set** of commitments
- $h^*(\sigma)$ measures (parallel) cost of best **complete plan** compatible with **partial plan** σ
- Definition of heuristics h^m can be transformed to provide informative lower bounds on $h^*(\sigma)$
- Good systems for **optimal parallel** and **temporal** planning will most likely rely on good **lower bounds** and suitable **branching schemes**
- Most planners can be understood along these two dimensions ...
- Stay tuned ...

Summary

- Planning as **general problem solving**
- Focus on **Heuristic Search Planning**
- **Heuristics** is key notion; **Branching** is key as well
- **Challenges**
  - richer action models (uncertainty, feedback)
  - richer temporal model (durations, parallelism)
  - better performance (heuristics, branching, ...)
- Important to keep certain **distinctions** in mind
  - optimal vs. suboptimal
  - sequential vs. parallel