Planning as Heuristic Search: New Results

Blai Bonet and Héctor Geffner
Universidad Simón Bolívar
Caracas, Venezuela

HSP: Heuristic Search Planner

- HSP is based on a simple idea:
  - derive **heuristic** from the representation
  - use **heuristic** to guide the search
- **Claim to fame:** only non-Graphplan based planner at AIPS98 Contest, where it did reasonably well

Talk today:

- present HSP and recent variation called HSPr
- relate "Planning as Heuristic Search" and Graphplan
- discuss limitations and challenges of Planning as HS

HSP: the State-space

The state-space defined by Strips problem $P = \langle Init, Goal, Ops, Atoms \rangle$ is such that

- the states $s \in S$ are collections of atoms
- the initial state s_0 is $Init$
- the goal states $s \in S_G$ are such that $Goal \subseteq s$
- 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 $s' = s - Del(op) + Add(op)$

HSP searches this space with an **heuristic** $h(s)$ that is extracted **automatically** from the problem description P

HSP: the heuristic

- the heuristic $h(s)$ provides an estimate of the cost from s to the goal
- $h(s)$ determined by computing the estimated costs $g(p, s)$ of achieving each **atom** p **from** s
- $h(s)$ defined then as $h(s) = \sum_{p \in G} g(p, s)$
- costs $g(p, s)$ computed by 'shortest-path' algorithm over graph connecting atoms with their preconditions; **delete lists are ignored**

Computation of the heuristic (details)

- Initialize costs $g(p,s)$ as

  $g(p,s) = 0$ if p is true in s, and $g(p,s) = \infty$ otherwise

- Do 'forward chaining' and when p asserted by op update

$$g(p,s) := \min[g(p,s)\ ,\ 1 + g(Prec(op), s)]$$

- Set $h(s) = g(Goals, s)$ when no more changes

– Two options for defininining $g(Set, s)$

  - $g(Set, s) = \sum_{p \in Set} g(p,s)$ (additive h), or
  - $g(Set, s) = \max_{p \in Set} g(p,s)$ (max h)

– HSP uses *additive* definition

HSP: search algorithm

- HSP uses Hill-Climbing with a few enhancements like limited number of 'plateau' moves, restarts, and memory
- Choice made from several considerations:
  1. very large state-spaces
  2. slow node generation/evaluation
  3. non-admissible but informative h
  4. many paths leading to same states

Some results from AIPS98 (McDermott)

Rnd	Planner	Time	Solved	Fastest	Shortest
1	BBOX	1.49	63	16	55
	HSP	35.48	82	19	61
	IPP	7.40	63	29	49
	STAN	55.41	64	24	47
2	BBOX	2.46	8	3	6
	HSP	25.87	9	1	5
	IPP	17.37	11	3	8
	STAN	1.33	7	5	4

- HSP strong in number of problems solved, but weak in solution times (and often in quality of solutions)
- main bottleneck is computation of heuristic $h(s)$ from scratch at every new state

HSPr: A regression heuristic planner

- HSPr addresses the main bottleneck of HSP; the computation of the costs $g(p,s)$ from scratch for **every new state** for determining the heuristic

$$h(s) = \sum_{p \in G} g(p,s)$$

- HSPr computes the costs $g(p, s_0)$ **only once** from s_0, and uses them to guide a **regression search**
- The heuristic $h_r(s)$ in the regression defined as

$$h_r(s) = \sum_{p \in s} g(p, s_0)$$

- Result is 5–10 faster node evaluation

HSPr: Additional features

1. HSPr uses more systematic algorithm ('Greedy' Best-First Search) that exploits faster node evaluation
2. HSPr uses also **mutex** idea from Graphplan to prune unreachable states

   Set of pairs M is a **mutex set** *iff for all pairs $\langle p, q\rangle$ in M:*

   1. *p and q are not both true in s_0, and*
   2. *the actions that add p, either delete q, or do not add q and have some precond r, $\langle r, q\rangle \in M$*
   3. *similar for q ...*

– There is a **largest mutex set** M^*
– HSPr computes an approximation of M^*
– Definition both **weaker** and **stronger** than Graphplan

Results: Logistic problems (Kautz & Selman)

	Time				Steps			
Prob	HSPr	HSP	BBOX	TLP	HSPr	HSP	BBOX	TLP
01	0.09	0.63	0.57	0.26	27	39	25	25
02	0.08	0.48	95.97	0.28	28	31	31	27
03	0.08	0.47	98.99	0.24	29	27	28	27
04	0.23	0.95	130.74	1.37	67	58	71	51
05	0.19	1.12	231.93	1.10	51	53	69	42
06	0.33	1.51	321.27	1.91	69	60	82	51
07	1.12	—	264.04	5.54	81	—	96	70
08	1.49	9.43	317.42	6.84	82	170	110	70
09	0.76	—	1609.45	3.79	77	—	121	70
10	0.99	4.77	84.04	2.42	46	109	71	41
11	0.64	1.61	137.93	2.24	54	47	68	46
12	0.43	1.30	136.22	1.93	41	36	49	38
13	1.26	5.78	165.84	6.54	74	102	85	66
14	1.46	8.48	77.74	9.34	82	141	89	73
15	1.10	3.53	424.36	5.36	69	76	91	63

Results: Logistic problems (Kautz & Selman)

	Time				Steps			
Prob	HSPr	HSP	BBOX	TLP	HSPr	HSP	BBOX	TLP
16	0.34	1.09	926.96	1.14	44	43	85	39
17	0.36	—	758.47	1.24	48	—	83	43
18	2.36	5.88	152.35	9.27	56	57	105	46
19	0.74	2.97	149.22	2.66	50	68	78	45
20	1.77	7.46	538.22	10.18	99	144	113	89
21	1.51	5.09	190.49	6.83	69	81	87	59
22	1.22	4.12	846.84	6.40	87	99	111	75
23	0.95	—	173.93	4.69	70	—	85	62
24	1.05	5.00	74.83	4.71	73	113	87	64
25	1.04	—	73.99	4.09	67	—	84	57
26	0.80	3.66	233.40	3.64	52	77	80	55
27	1.08	4.45	145.16	5.52	76	115	97	70
28	2.96	—	867.34	14.53	88	—	118	74
29	2.40	8.40	89.51	5.99	50	105	84	45
30	0.89	—	495.37	3.42	52	—	92	51

Results (cont'd)

- Over `logistics` problems:
  – HSPr better and more robust than HSP
  – two orders of magnitude faster than Blackbox and better sequential plans
  – competitive with highly tuned TLPlan
- Over `blocks world`, HSPr sometimes weaker than HSP
- HSP heuristic not always as informative for regression

		HSPr		HSP	
Problem	Blocks	Time	Steps	Time	Steps
bw-01	7	0.07	6	0.16	6
bw-02	9	0.15	6	0.59	10
bw-03	11	0.33	10	0.77	12
bw-04	15	8.11	18	5.88	19
bw-05	19	6.96	18	15.80	28
bw-06	8	0.12	19	0.42	12
bw-07	10	1.10	13	0.65	12
bw-08	12	3.73	14	1.97	18
bw-09	14	–	–	3.83	17
bw-10	16	–	–	12.63	20

Planning as Heuristic Search and Graphplan

- HSPr and Graphplan work in two phases. In the second, they both do regression search; in the first they do 'forward' reasoning:
  - HSPr computes atom costs $g(p, s_0)$
  - Graphplan computes planning graph

- **Claim:** main function of graph is **the representation of a heuristic** $h_G(s) = i$, where i is the index of first layer that contains s with no mutex

– h_G is refined version of 'max' heuristic

$$h_{max}(s) = \max_{p \in s} g(p, s_0)$$

– h used in HSPr is 'additive' heuristic

$$h_r(s) = \sum_{p \in s} g(p, s_0)$$

Graphplan as a Heuristic Search Planner

1. **Planning Graph** encodes admissible heuristic h_G

   $h_G(s)$ is index of first level that contains s with no mutex and no memoization

2. **Search algorithm** is IDA* (not blind ID)

   sum *of the accumulated and estimated cost $h_G(n)$ is used to prune nodes n*

3. **Memoizations** are updates on h_G, resulting in version of IDA* known as MREC (Sen & Bagchi 89)

   in MREC, $h(n)$ updated and cached after search below n completes with no solution

Why then is Graphplan better than traditional planners?

because traditional planners do **blind** search

Summary; Challenges

- **Planning as heuristic search** is a useful and practical framework for planning
- Progress needed along a number of dimensions
  - better domain-independent heuristics
  - faster node generation
  - better use of memory
  - better modeling languages

- A lot of work done in the area, and a lot to do ...