Visit-All domain

November 22, 2010

1 The Problem with Multiple Conflicting Goals

The heuristics used in state-of-the-art (satisficing) planners are a decade old and are based on the delete-relaxation.[1] Several heuristics that take deletes into account have been formulated but they haven't been shown to be cost-effective. One problem with delete-relaxation heuristics that aproximates h^+, appears in instances with multiple 'conflicting' goals. In these cases, that are very common, progress towards one goal means moving away from other goals. Such instances produce large plateaus where the heuristic is almost useless. Indeed, in some cases, state-of-the-art heuristics are no better than *heuristics that ignore the problem structure completely* and just count, for example, the *number of unachievable goals.*

As an illustration of this, consider the Visit-All domain where an agent in the middle of a square grid $n \times n$ must visit all the cells in the grid. Solving optimally the delete relaxation h^+ gives the exact goal distance as long as there exists a hamiltonian path visiting every cell. Recall that in a $1 \times n$ grid, no hamiltonian path exists. This is an extremely simple problem to solve non-optimally and a version of this problem is related to one of the domains used in the Conformant track of the 2008 IPC.[2] Table 1 shows results for several planners over this domain. Interestingly, the planner that does best in this domain is based on a greedy best-first search using *the number of unachieved goals heuristic* ($hgoals$), that manages to solve the larger 50×50 instance in 122 seconds. The greedy best search using the *additive heuristic* does much worse, and can't even solve the 15×15 instance. FF, like LAMA when run with the FF heuristic only, don't do much better and don't solve the 20×20 instance. On the other hand, LAMA, and the version of LAMA that uses the *landmark heuristic* only, do quite well, and fail on the 35×35 instance due to 'grounding errors'. The performance of LAMA, however, is not surprising as the landmark heuristic in this case is quite close to the 'number of unachieved goals' heuristic.

We emphasize that domains like Visit-All show that some of the problems that are difficult for 'pure' heuristic search planners are not hard at all, they are just the result of multiple easy but conflicting goals that can often be achieved trivially, once they

[1] The exception is the landmark heuristic in LAMA, discussed below.

[2] We are referring to the Dispose domain where an agent has to pick up objects in a grid without knowing their locations, and thus has to attempt a pick up action in each cell. In this domain, a conformant planner guided by the 'number of unachieved goals' heuristic did much better than the classical FF planner over a suitable translation. Table 1 explains how this can happen.

are serialized. The use of delete-relaxation heuristics to appraise the cost of achieving all goals together runs into a situation resembling Buridan's ass: where a hungry and thirsty donkey, placed between a bundle of hay and a pail of water, dies of hunger and thirst for not finding a reason to choose one over the other.

	LAMA		LAMA-ff		LAMA-lm		FF		GBFS-hadd		GBFS-hgoals	
I	time	cost	time	cost	time	cost	time	cost	time	cost	time	cost
5	0.15	26	0.22	36	0.17	26	0.05	38	0.02	98	0.01	304
10	0.43	128	0.55	352	0.40	111	0.16	177	7.15	883	0.04	104
15	1.39	270	118.60	1,258	1.00	271	15.38	428	M	M	0.32	276
20	3.42	493	M	M	2.92	453	M	M	T	T	1.14	417
25	10.32	757	M	M	5.98	732	M	M	T	T	3.58	660
30	26.38	1,149	M	M	14.10	1,082	M	M	T	T	9.07	956
35	R	R	R	R	R	R	M	M	T	T	20.65	1330
40	R	R	R	R	R	R	M	M	T	T	47.43	1920
45	R	R	R	R	R	R	M	M	T	T	74.04	2140
50	R	R	R	R	R	R	M	R	M	M	122.19	2550

Table 1: Planners performance on the Visit-all domain. The planners are *LAMA* and versions of it using only the FF and Landmark heuristic respectively, *FF*, two greedy best first-search planners using the additive and number of unachieved goal heuristics respectively. The grids are $n \times n$, with the number n reported in each row. For the unsolved problems, M, T, and R stand for memory-out (2GB), time-out (30min.), and grounding error respectively.