Research Review Planning

The assignment leaves us a lot of room to pick the specific developments in this field that we want to cover. I chose to look at those as an extension to what we have learned so far, specifically about search.

- Combining search and inference
- Introducing structure to describe problems domain independently
- Deriving heuristics from a general problem description

Combining search and inference

So far we have learned a lot about searching a problem space. An emphasis has been on cutting down the search spaces as early as possible to limit the consequence of combinatorial explosions.

We have seen this in action already in the Sudoku assignment from before. There we applied inference to eliminate a large part of the problem space. Sometimes this was already good enough to come to a solution, sometimes search still had to be applied. The implementation we used applied inference and search interleaving.

For the specific purpose of planning this principle is also applied in the Graphplan[1] algorithm.

Instead of building a complete search graph from logic, Graphplan incrementally builds one level of a search tree at a time and then tries to extract a solution that satisfies all goals from this level. If no solution is found, the search tree is expanded for another level until either a solution is found or the tree levels off and it becomes evident that no solution can be found.

The tree levels off when the next expansion does not produce new - so far unseen - states, that have the potential to allow for new results.

Introducing structure to describe domain independent heuristics

So far when we used heuristics to guide a search process, e.g. for A* search, we used our understanding of the domain, e.g. the euclidean distance of the exists position relative to our own position in the Pacman maze.

While this did work, it makes us depend on domain knowledge and makes it harder to devise general purpose planning algorithms.

[1] Avrim Blum, Merrick Furst (1995, 1997)

With the introduction of a formal description for problems we are decoupling the actual domain knowledge and the structure of a problem. There is still the need to understand the domain - as it needs to be translated into the generic structure -, but its use is limited to that purpose and also provides the proper vocabular to discuss the problem domain with a domain expert.

Something like the Problem Domain Description Language (PDDL) [2] allows a domain problem to be expressed in a planning language.

Deriving heuristics from a general problem description

Now having the problem described in a uniform fashion as understood by the solving algorithm, this allows for a stronger separation of solving logic and problem description [3] [4]. It also opens up new possibilities to derive heuristics.

The solving algorithm can now benefit from the domain semantics expressed as planing elements. For example preconditions of an action and effects when executing the actions. From there, using those preconditions and effects, it can be infered if actions are mutably exclusive and therefore the problem space can be adjusted accordingly.

In the same way heuristics can also be derived by solving a simpler problem. For example some or all preconditions (like in our programming exercise) can be dropped and then we can count the number of levels until we are able to satisfy our goals. In effect only solving the simpler issue, but given that we only shoot for an admissable heuristic (lower or equal estimation as compared to the actual cost) this tradeoff may pay off.

[2] Ghallab et al. (1998)
[3] Strips: Fikes and Nilsson (1971)
[4] General Problem Solver (GPS): Newell and Simon (1961)