Report on AI planning and search development

Means-ends analysis is one of the oldest ideas in AI and is still an important idea today. Means-ends analysis is to find a sequence of actions that, if carried out starting in the initial situation, would get to a situation that satisfies the goal description. The idea behind means-ends analysis is to build action sequences by continually adding actions whose add lists contain conjuncts that correspond to pieces of the goal-situation description. The bug of the idea is that action is hard to satisfy below 2 conditions: 1.The action is feasible and 2. The action's add list occurs in the goal description. The solution before is to make search states more complicated, by keeping track of a hierarchy of sub goals as well as a plan prefix.

Drew McDermott, in 1999, proposed another alternative: Keep search states simple, just sequences of actions and try harder to find feasible and relevant actions to add. However, instead of cautiously back-chaining in tiny steps, back-chain all the way to feasible actions at every search state. This is a new way of thinking about means ends analysis, in which an exhaustive sub goal analysis, based on greedy regression-match graphs, is repeated at each search state, rather than being spread over a number of search steps. It yields two benefits: a heuristic estimator of the number of actions required to get to a problem solution from the current state; and a set of actions that are good candidates for the next step to take. The resulting situation-space search algorithm searches many fewer states than traditional planners on a large class of problems, although it takes longer than usual per state (*A Heuristic Estimator for Means-Ends Analysis, Drew McDermott*).

In 2000, a family of heuristic search planners that based on a simple and general heuristic that assumes that action preconditions are independent are studied (*Planning as heuristic search, Blai Bonet, Héctor Geffner*). It studied three planer. 1,HSP uses the additive heuristic hadd to guide a hill-climbing search from the initial state to the goal. The hill climbing search is very simple: at every step, one of the best children is selected for expansion and the same process is repeated until the goal is reached. HSP is competitive with the best current planners over many domains. However, HSP is not an optimal planner, and what's worse—considering that optimality has not been a traditional concern in planning—the search algorithm in HSP is not complete. 2, HSP2 uses the additive heuristic hadd in a best-first search from the initial state to the goal. This best-first search keeps an Open and a Closed list of nodes as $A*$ but weights nodes by an evaluation function $f(n) = g(n) + W \cdot h(n)$, where g(n) is the accumulated cost, h(n) is the estimated cost to the goal. It tends to be faster and more robust than the hill-climbing HSP planner. 3, HSPr tries to avoid the bottleneck in both HSP and HSP2 in the computation of the heuristic from scratch in every new state. This takes more than 80% of the total time in both planners and makes the node generation rate very low. This problem could be solved by performing the search backward without recomputation to define heuristic. The benefit of the search scheme is that node generation will be 6–7 times faster. However, in many problems the new search scheme does not help, and in several cases, it actually hurts.

In 2006, Malte Helmert proposed Fast Downward (*The Fast Downward Planning System*, Malte Helmert). Like HSP and FF, Fast Downward is a progression planner, searching the space of world states of a planning task in the forward direction. However, unlike other PDDL planning systems, Fast Downward does not use the propositional PDDL representation of a planning task directly. Instead, the input is first translated into an alternative representation called multivalued planning tasks, which makes many of the implicit constraints of a propositional planning task explicit. Exploiting this alternative representation, Fast Downward uses hierarchical decompositions of planning tasks for computing its heuristic function, called the causal graph heuristic, which is very different from traditional HSP-like heuristics based on ignoring negative interactions of operators. The causal graph heuristic is the centre piece of Fast Downward's heuristic search engine. It estimates the cost of reaching the goal from a given search state by solving a number of subproblems of the planning task which are derived by looking at small "windows" of the (pruned) causal graph. The planner has shown excellent performance: The original implementation of the causal graph heuristic, plugged into a standard best-first search algorithm, outperformed the previous champions in that area, FF and LPG (Gerevini, Saetti, & Serina, 2003), on the set of STRIPS benchmarks from the first three international planning competitions (Helmert, 2004). Fast Downward itself followed in the footsteps of FF and LPG by winning the propositional, non-optimizing track of the 4th International Planning Competition at ICAPS 2004.